The Resource Journal For IBM PC Programmers

# PROGRAMMER'S Journal

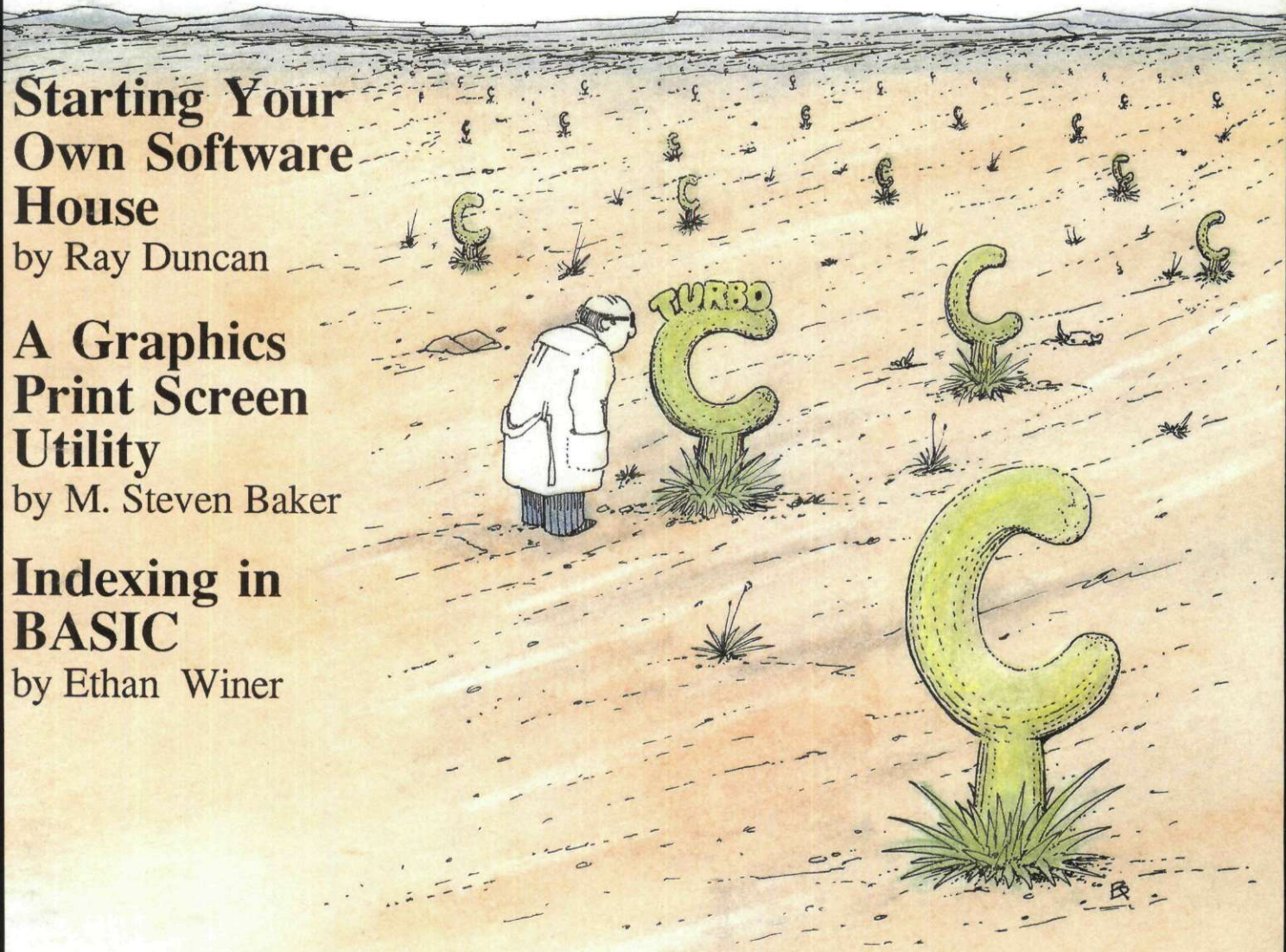
MARCH/APRIL 1987
VOLUME 5.2
$3.95

# TURBO C:
## A First Look
by Michael Abrash

## Starting Your Own Software House
by Ray Duncan

## A Graphics Print Screen Utility
by M. Steven Baker

## Indexing in BASIC
by Ethan Winer




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# ACTOR WILL DO WINDOWS.

If you've been wondering how you're going to explore Microsoft® Windows and then work it into your programs, what you need has just arrived.

ACTOR!™

ACTOR is a new programming system, and there are two reasons why it's so good with Windows.

First, ACTOR is an ideal programming environment. It shows you all your work through Windows. Any part of the program you're writing, its output as it runs, error diagnostics and tracing, catalogs of routines, and, in fact, just about anything you like—they can all be visible on the screen and accessible at the same time, in their own windows.

So it's easier to see what you're doing.

Second, ACTOR is an object-oriented programming language. Everything in an ACTOR program is a self-contained package with its own data and procedures. That is, an object. And the objects pass messages to each other while the program runs.

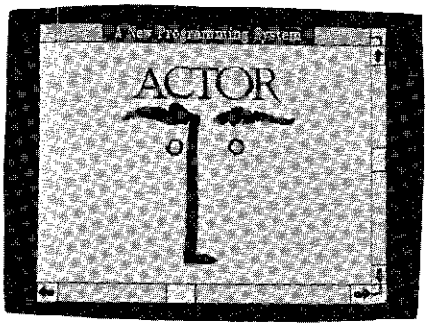
With ACTOR, you don't write lists of instructions. Instead, you define what classes of objects do. Which is especially easy because they inherit most of their behavior from other classes. That means there's less for you to worry about.

And when you create an object, you just choose its name and starting values. After that it pretty much takes care of itself.

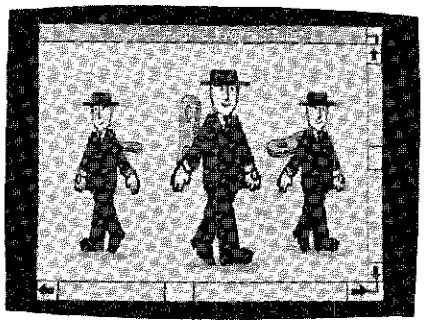
What's important here is that windows are objects like everything else in the language. You make as many as you need, and they do what they're supposed to, mostly on their own.

So it's natural and easy to put windows in your application programs. And users will benefit from Windows when your programs run, as much as you do when you write them.

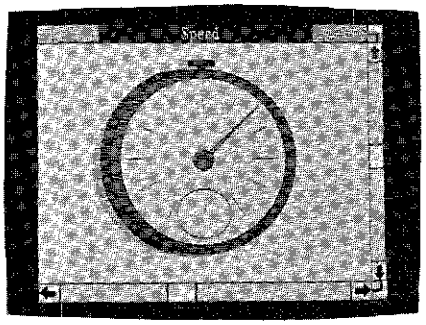
To order, just send in the coupon. Or call **312-491-2370** for more information. You'll see that ACTOR will do Windows, and more.



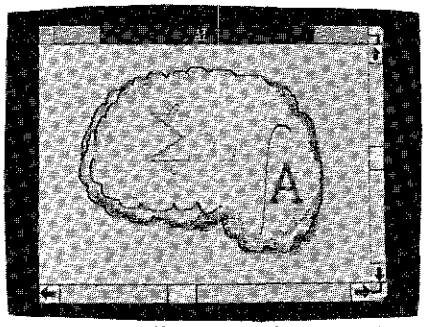

ACTOR™ is a new, interactive programming system, the first with Microsoft® Windows. You have Windows when you write a program, and users have Windows when they run it.



Moreover, ACTOR is a new, object-oriented programming language. Winding objects up and turning them loose is a lot easier, more productive, and more fun than old fashioned programming.




Thanks to a new method of "incremental garbage collection," ACTOR never has to slow down to clean up memory like other advanced languages. You can even use it for real-time control. It's that fast.



ACTOR offers all the features of an ideal artificial intelligence language, but in a familiar, Pascal-style syntax. Which makes artificial intelligence programming easier. And what could be smarter than that?

---

Please send me more information about ACTOR.

Name_____

Address_____

City_____ State_____ Zip_____

Daytime phone_____

☐ Check or money order enclosed. ☐ Bill my VISA. ☐ Bill my MasterCard.

Card number_____ Exp. date_____

☐ I'd like to place an order while I'm at it. (ACTOR is not copy protected and has a 60-day money back guarantee. Educational pricing is available. ACTOR requires an IBM PC/XT or AT or equivalent. 640K and hard disk is recommended.)

☐ ___ copies of ACTOR with Windows runtime support at $495 each. $_____

☐ ___ copies of ACTOR, including Windows complete with MS-DOS Executive, at $550 each. $_____

■ Please add $5 for shipping within the U.S., $30 overseas. $_____

■ Total. $_____


The Whitewater Group®
Technology Innovation Center
906 University Place, Evanston, IL 60201


Circle Number 73 on the Reader Service Card


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# EVERYTHING THE C PROGRAMMER NEEDS

## Eco-C88 C Compiler:

A full, professional C compiler with many ANSI enhancements at an unbelievably low price.

* Prototyping, enum, void data types, plus structure passing & assignment * All operators and data types * Over 200 library functions * cc and mini-make for easy use * 8087 support * ASM or OBJ output * Lint-like tiered error messages * Fast code * CED editor (edit-compile-link from within the editor) * Expanded user's manual * Not copy protected * All for only $59.95!

## Eco-C88 Flexi-Graph Graphics Package

Everything you need to write dramatic graphics effects into your Eco-C88 C programs.

* EGA, CGA, and Z100 support * Over 100 graphics functions (many are PLOT-10 compatible) * Most assembler support routines are outside small model code-data * Write thru BIOS (for compatibility) or to memory (for speed) * Graphics function help from CED editor * World, pixel or turtle color graphics modes * 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash, and fonts * Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual * Only $39.95

## Eco-C88 Windowing Library

Use this library to build pop-up windows, help windows, selection menus, special effects—anywhere you need an attention getter.

* CGA and EGA support * Control any program that goes through the BIOS * Use up to 255 windows * No special window commands — use plain old printf( ) to write to a window * Resize and move windows * Custom window titles and borders * Can be used with ANSI device driver * Most window code-data are outside small model * User's manual and examples for only $29.95

## Ecosoft Librarian

Combine your modules, functions, and subroutines into your own library for easy link commands. Compatible with any standard MSDOS OBJ files * Add, delete, and extract from a library * Get table of contents or index of a library * Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files * Complete with user's manual for only $29.95

## Developer's Library

Contains the source code for for all library functions, including the transcedentals, memfiles, and those written in assembler. $25.00 with order, $50.00 if ordered later. (Sold only to Eco-C88 owners.)

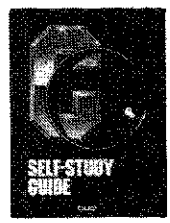
**Ecosoft Inc.**
6413 N. College Ave.
Indianapolis, IN 46220

*o-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 6K of memory, 2 disk drives and MSDOS 2.1 or later.*

## C Programming Guide

**(Purdum, Que Corp.)**

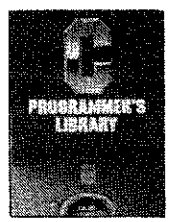
The second edition of the B. Dalton bestseller. Perfect for those just getting started with C. Includes discussion of many X3J11 ANSI Standards Committee recommendations. Many error messages from Eco-C88 give page references to this book. Price is $20.00 plus $2.00 shipping.




## C Self-Study Guide

**(Purdum, Que Corp.)**

Using a question-answer approach, this book is filled with shortcuts, tips, techniques, and traps to avoid when learning C. Price is $17.00 plus $2.00 for shipping.




## C Programmer's Library

**(Purdum, Leslie, Stegemoller, Que Corp.)**

Another B. Dalton bestseller. An intermediate C text for the programmer that wants to get the most from the language. Contains source code for many functions including an ISAM file handler. Price is $22.00 plus $2.00 for shipping.

To order, call or write:

**1-800-952-0472** (for orders) or **1-317-255-6476** (tech. info.)

ORDER FORM CLIP & MAIL TO:
Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

- ☐ Eco-C88 C Compiler $59.95 _____
- ☐ Eco-C88 Flexi-Graph Graphics Package $39.95 _____
- ☐ Eco-C88 Windowing Library $29.95 _____
- ☐ Ecosoft Librarian $29.95 _____
- ☐ Developer's Library $25.00 ($50.00 if not with order) _____
- ☐ C Programming Guide $20.00 _____
- ☐ C Self-Study Guide $17.00 _____
- ☐ C Programmer's Library $22.00 _____

SHIPPING $4.00
TOTAL (IND. RES. ADD 5% TAX) _____

PAYMENT: ☐ VISA ☐ MC ☐ AE ☐ CHECK
CARD# _____ EXPIR. DATE _____
NAME _____
ADDRESS _____
CITY _____ STATE _____
ZIP _____ PHONE _____

Number 8 on the Reader Service Card

**ECOSOFT**

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

Volume 5.2
March/April 1987

# Contents

4 **Note from the Editor**

6 **Letters**
Optimizing Turbo, memory resident code, and more

10 **PJ Forum:** DTACK GROUNDED, The Junk Mail Flyer *by Hal W. Hardenbergh*
Marketing plans and raucous laughter, HP, the 68030, the Transputer, and expert idiots

16 Starting Your Own Software House *by Ray Duncan*
The pros and cons and the preliminaries

22 **Redirections:** Pink Noise *by Frank D. Greco*
IBM's PL.8 compiler, monitors, parallel processing, animation on the PC, the latest on laser printers

26 **ASM:** A Graphics Print Screen Utility: a Simple Terminate and Stay Resident Program *by M. Steven Baker*
Graphics screen dumps to an Okidata printer

34 **ASM:** Coding Guidelines for Resident Programs *by Graham Pearson*
Some sample memory resident code

40 **On Graphics:** Parallel Processing with the EGA *by Michael Abrash*
The state of PC graphics, starting a tour of the EGA, and some reader challenges

48 **C:** Portability in Practice and Using Header Files *by William James Hunt*
Practical problems in writing portable C code and a technique for keeping track of header files

50 **C:** The Proposed ANSI C Language Standard, Part 6 *by Rex Jaeschke*
Function prototypes

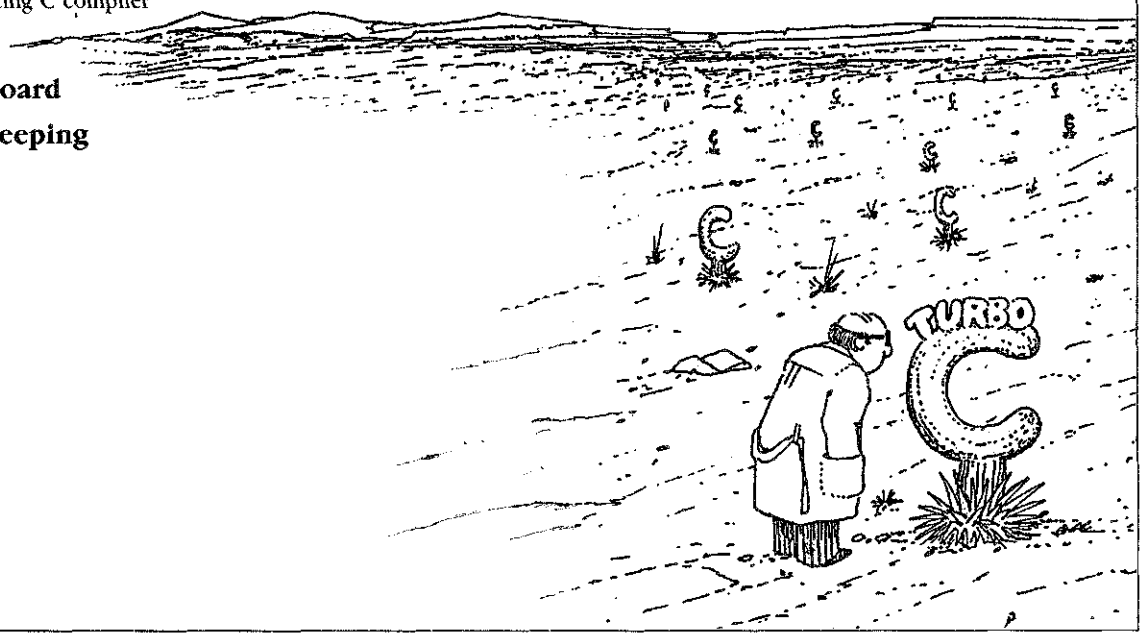
54 **BASIC:** Indexing in BASIC *by Ethan Winer*
Indexing techniques to speed BASIC file access

62 **Review:** Turbo C: A First Look *by Michael Abrash*
An interesting C compiler

68 **Misc.**

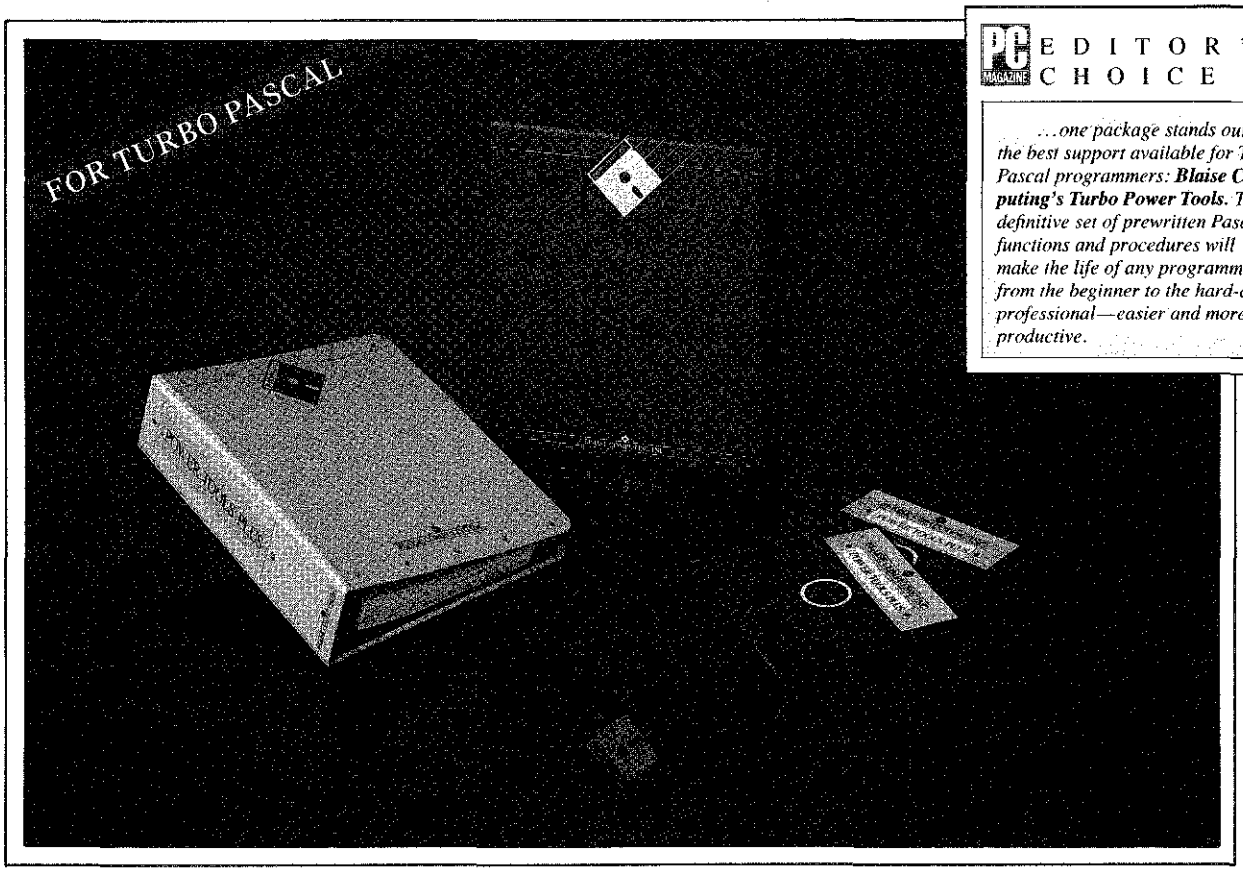
71 **PJ Billboard**

72 **Housekeeping**






PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




**PC MAGAZINE EDITOR'S CHOICE**

*...one package stands out as the best support available for Turbo Pascal programmers: **Blaise Computing's Turbo Power Tools.** This definitive set of prewritten Pascal functions and procedures will make the life of any programmer—from the beginner to the hard-core professional—easier and more productive.*

# Power Tools Plus™

## Window Routines. Memory Resident Routines. Routinely.

### ANOTHER PLUS FROM BLAISE COMPUTING

The best just got better! Turbo POWER TOOLS, acclaimed as the best programmer support package for Turbo Pascal, now has even more functions, more detailed documentation and more sample programs.

### NO SECRETS

Turbo POWER TOOLS PLUS is crafted so that the source is efficient, readable and easy to modify. We don't keep secrets! We tell you exactly how windows are managed, how interrupt service routines can be written in Turbo Pascal, and how to write memory resident programs that can even access the disk. Maybe you've heard of some undocumented DOS features that resident programs use to weave their magic. Turbo POWER TOOLS PLUS documents these features and lets you make your own magic!

*Here's just part of the PLUS in Turbo POWER TOOLS PLUS:*

- **WINDOWS** that are stackable, removable, with optional borders and a cursor memory.
- **FAST DIRECT VIDEO ACCESS** for efficiency.
- **SCREEN HANDLING** including multiple monitor and EGA 43-line support.
- **POP-UP MENUS** which are flexible, efficient and easy to use, giving your applications that polished look.
- **INTERRUPT SERVICE ROUTINES** that can be written in Turbo Pascal without the need for assembly language or inline code.
- **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple procedure calls, you can "schedule" a Turbo Pascal procedure to execute either when a "hot key" is pressed, or at a specified time.
- **PROGRAM CONTROL ROUTINES** allow you to run other programs from Turbo Pascal, and even execute DOS commands.
- **MEMORY MANAGEMENT** allows you to monitor, allocate and free DOS-controlled memory.
- **DIRECTORY AND FILE HANDLING** support to let you take advantage of the newer features of DOS including networking.
- **STRING** procedures allowing powerful translation and conversion capabilities.
- **FULL SOURCE CODE** for all included routines, sample programs and utilities.
- **DOCUMENTATION, TECHNICAL SUPPORT** and attention to detail that have distinguished Blaise Computing over the years.

***Turbo POWER TOOLS PLUS** supports Turbo Pascal Version 2.0 and later and is just **$99.95.***

*Another quality product from Blaise Computing: **Turbo ASYNCH PLUS™***

*A new package which provides the crucial core of hardware interrupt support needed to build applications that communicate. ASYNCH PLUS offers simultaneous buffered input and output to both COM ports at speeds up to 9600 baud. The XON/XOFF protocol is supported. Now it also includes the "XMODEM" file-transfer protocol and support for Hayes compatible modems.*

*The underlying functions of Turbo ASYNCH PLUS are carefully crafted in assembler for efficiency and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The high level function are all written in Turbo Pascal in the same style and format as Turbo POWER TOOLS PLUS. All source code is included for just $99.95.*

## BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

**ORDER TOLL-FREE 800-227-8087**

**Calif. residents call (415) 540-5441**

YES, send me the PLUS I need! Enclosed is $_____ for
☐ Turbo POWER TOOLS PLUS ☐ Turbo ASYNCH PLUS ☐ OTHER _____ (CA residents add 6½% Sales Tax. All domestic orders add $10.00 for Federal Express shipping.)

Name: _____ Phone: (____) _____

Shipping Address: _____

City: _____ State: _____ Zip: _____

VISA or MC #: _____ Exp. Date: _____

Number 2 on the Reader Service Card

Reprinted from PC Magazine, June 10, 1986 Copyright © 1986 Ziff-Davis Publishing Company

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal For IBM PC Programmers
PROGRAMMER'S Journal

**Editor**
Robert Keller

**Managing Editor**
Erin O'Connor

**Technical Editor**
M. Steven Baker

**Advertising & Production**
Ralph Bentley

**Advertising & Promotion**
Robert Tappan

**Illustration**
Jim Carpenter

**Circulation**
Gary Hoyer

**Publisher**
Elizabeth Oakley

**Founder**
Greg Estes

**Contributing Editors**
Frank D. Greco, Tom Swan

Contributing Writers
Michael Abrash
M. Steven Baker
Ray Duncan
Frank D. Greco
Hal W. Hardenbergh
William J. Hunt
Rex Jaeschke
Graham Pearson
Ethan Winer

**Business and Editorial Offices:**
Programmer's Journal
29 West 29th Ave., Suite 5
Eugene, OR 97405
(503) 484-2162

**Subscription Inquiries:**
Programmer's Journal
P.O. Box 3000
Dept. EE
Denville, NJ 07834
(503) 484-2162

Subscriptions:
1 year (6 issues) $24 U.S. ($30 in Canada and $40 elsewhere) 2 years $44 U.S. ($55 in Canada and $75 elsewhere) 3 years $64 U.S. ($80 in Canada and $110 elsewhere)

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.
Advertising: For ad rates, deadlines, and placement, contact Ralph Bentley, Robin Tappan, or Gary Hoyer (503) 484-2162, or write to Programmer's Journal, P.O. Box 30660, Eugene, OR 97403.
IBM and Personal Computer are trademarks of International Business Machines; dBASE II and dBASE III are trademarks of Ashton-Tate.
Programmer's Journal (ISSN 0747-5861) is published bi-monthly by Oakley Publishing Company. Second-class postage paid at Eugene, OR and additional mailing offices.
POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 3000, Dept EE., Denville, NJ 07834.
©1987 OAKLEY PUBLISHING COMPANY

# NOTE FROM THE EDITOR

**Ray Duncan**
Ray Duncan starts a series in this issue on the steps involved in becoming a small software vendor. In subsequent installments, he plans to cover advertising strategies, technical support, and packaging and documentation. (Unless, of course, you, the readers, influence his direction along the way. Ray's writing always seems to include a lot of interplay with readers.)

**On Graphics**
Also in this issue, Michael Abrash continues his column on graphics. He examines Comdex and the state of PC graphics from the developer's point of view, begins a detailed tour of the EGA, and ends with a couple of challenges to the readers.

**Turbo C**
In this issue we're bringing you a first look at Borland's long-rumored Turbo C. Michael Abrash spent an afternoon with the developers at Borland's company headquarters and came away impressed.

Microsoft is rumored to be using Quick-C, a C compiler built along the lines of QuickBASIC, in-house. Perhaps Borland's release will lead to a release from Microsoft.

**Crowded**
We're running out of room. If I'd had my way, this issue would have had more from DTACK, a complete communications package callable from BASIC by Steve Baker, more C code from William Hunt, Paul Barkley's Consulting column, Part 2 of Frank Greco's Unix book review, Frank's interview with the president of Metagraphics, and more. We need more subscribers so that we can bring you more pages.

And, you programmers, take the time to share your hard-won knowledge with *PJ*'s readers. Put a paragraph on the front of some code you like and send it to us.

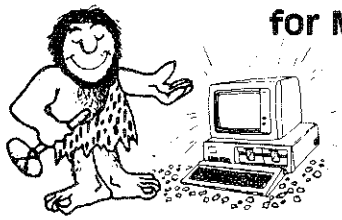
**Peer Review**
You'll notice that the Peer Review Card has been redesigned. We want you to know that we read those cards diligently and pass on what you say to the authors, whether your ranking is "1+++" or a little figure eight tipped on its side, whatever that means.

—Robert □


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Professional Programming Products

## for Microsoft C, PASCAL, FORTRAN, and Assembly Language



# FREE!

# PC-WRITE™ text editor

**PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!**
**NO ROYALTIES REQUIRED**

### ASMLIB
### The Programmer's Library

- A linkable library with over 210 assembly language subroutines which are directly CALLable from Microsoft™ Assembler, C, PASCAL, and FORTRAN.
- Virtual disk file handling.
- Text WINDOWing allows up to 64 overlapping windows.
- Int. driven asynch. support.
- Graphics on EGA, herc. and CGA.
- Floating point math and trig outlines with 8087 support.
- Installable keyboard activated programs are easily written with ASMLIB's special functions.
- Plus much, much more.
- Supplied with complete source code.

**Only $149.00 Complete**

### asmTREE
### The Programmer's B+Tree Data File Management System

- A complete single/multiuser database development system written in assembly language and CALLable directly from Microsoft™ Assembler, C, FORTRAN, PASCAL, and LATTICE™ C programs.
- Up to 256 users.
- Up to 256 index and data files.
- Multiple key types.
- Multiple indices per index file.
- Duplicate and variable length keys.
- Read DBASE data files directly.
- Plus much, much more
- Supplied with complete source code.

**Only $395.00 Complete**

NEW

### FPLIB
### FPLIB™-IEEE Floating Point for REALIA™ COBOL!

- A full IEEE floating point math package for your REALIA COBOL programs.
- 8087 numeric co-processor support and EMULATION. Switching from emulation to true 8087 instructions is done automatically at RUN time - transparent to the user!
- A full set of both arithmetic and TRIGONOMETRY functions have been provided.
- Conversion routines allow your REALIA program to pass data to and from FPLIB by either using DISPLAY types or COMP-5 data types.
- FREE Updates for 1 year.
- SOURCE CODE available.

**Only $149.00 Complete**

NEW

### Turbo.ASM
### For TURBO Users!
### Turbo.ASM™ (turbo dot a.s.m.)

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for the Turbo user who wants to get into Assembly Language.
- Outlines 4 different ways to add assembly language routines to Turbo programs without effecting your code space.
- Fully explains internal workings of Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- NO ROYALTIES required.

**Only $99.95 Complete**

**B C ASSOCIATES**
3261 No. Harbor Blvd., Suite B
Fullerton, CA 92635
**1-800-262-8010**
in Calif. Call
**(714) 526-5151**

Call for information on complete line of SimpleNET networking products.

**Outside CA, call TOLL FREE 1-800-262-8010**

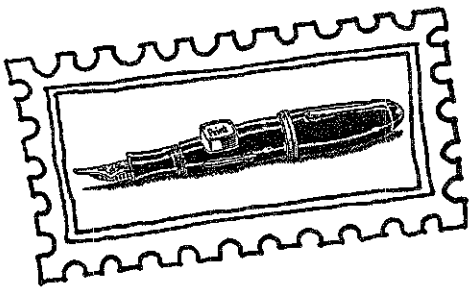
**USE YOUR VISA OR MASTERCHARGE -**

All prices include UPS shipping within continental United States. Outside U.S. please add $10 per package. Calif. residents please add 6.5% sales tax.

Number 36 on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



**An Alternative to Undocumented 34H**

M. Steven Baker's article "Terminate and Stay Resident" in the November/December 1986, issue (*PJ*, 4.6) is indeed one of the most informative on this subject that I have come across, particularly in its discussion of undocumented DOS TSR support. There is, however (as Mr. Baker notes), always a danger in using undocumented features (remember " . " in place of "rem" for bat files in DOS < 3. versions), and if it is at all possible, one should refrain from their use even if it is unlikely that future relevant versions may change them. I do wonder how they will appear in DOS 4.0, which has appeared in the U.K. and is about to here.

In any event, the purpose of this letter is to note an alternative to the use of the DOS undocumented function 34H (the "H" was missing in the article), the so-called DOS_CRITICAL or INDOS function. One aspect of a TSR program is that after it is loaded and activated, only the CS segment register at the initial entry point is redefined; the others maintain the values held when the TSR was activated. Thus, if the first thing the resident program does is to sample the segment and some other registers, much useful information on the program that was active when the TSR was activated can be deduced. The basic assumption made is that the TSR was *initially* loaded directly from DOS and not from another application—a reasonable assumption.

Listed below is the Assembler code included at the entry point to the resident program:

```
resident:                          ;start resident code.
        assume cs:cseg
        _save_entry_regs           ;save entry registers
        assume  ds:cseg,es:cseg    ;now can assign regs
        mov     ax,cs              ;initialize registers
        mov     ds,ax
        mo      es,ax
        cmp     cs:old_ss,ax       ;see if ss below this CS
        jae     is_ok              ;is a program ss
        cmp     bp,0               ;if 0 is DOS
        je      is_ok              ;OK if DOS
        call    set_res_page
        $putchar        7
        $putstring cs:iret0,cs:
        . . . .
        . . . .
is_ok:                             ;OK to continue
        . . . .
        . . . .
```

"save_entry_regs" is a macro that saves all the registers at the entry point in a data section in the current CS segment. The next three lines initialize the DS and ES registers (I use either my own stack or the system stack, depending on other factors) and then proceed to test the SS segment register at entry. If it was less than the current CS segment register, that indicates that the active program from which the TSR was initiated was either DOS or a program that was using the DOS stack. Which can be inferred by testing the BP register. If the BP register is 0, the active program was DOS; if not 0, the active program was another program using the DOS stack.

If the active program was another program using the DOS stack, you cannot safely make DOS function calls, and unless the TSR is benign with respect to DOS function calls, you do best to exit it.

If BP is 0 (meaning that the TSR was activated from DOS), you can go ahead and make DOS calls so long as you exit the TSR via DOS function 4CH.

If the TSR was activated from another program *not* using the DOS stack, you can go ahead and make DOS function calls and return to the program from which the TSR was activated. I have tested this with a variety of programs and found no problems. I did find one program that appears in some very special circumstances to occasionally use the DOS stack, and when I honor that use, no problems occur there, either.

Just for the record, the next three lines after "_save_entry_regs" call a procedure that tests the video page used by the program from which the TSR was activated and sets the next video page for use by the TSR, sound a beep ("$putchar 7" is a macro), and display a message (yet another macro) warning that the TSR was called from a program using the DOS stack and offering a series of alternatives.

I have run a series of tests using the code above in conjunction with the undocumented function 34H and found that whenever the SS segment register was less than the CS of the TSR, function 34H returned the value of 1, whether or not the TSR was activated from DOS or from another program. The programs that seem to use the DOS stack are, for the most part, *interpreters*. I have PROLOG86, SNOBOL4, IQLISP, and PC SCHEME, and they all use the DOS stack. BASIC, not surprisingly, does not.

Having gone through this, I must now admit that in general I find myself operating in the same way as Mr. Baker. As a rule, I do not use memory resident programs except for very special circumstances. It is not so much that I don't find what TSRs offer useful, but rather that I can avail myself of what they have to offer in simpler and potentially less damaging ways.

Any editor worthy of its salt allows you to edit multiple files, and you can always set up a file in a window for note taking as well as one with useful info in it. Again, most editors have a calculator built in (including Hex), or you can write one using an editor's macro facilities, or just have one sitting on your desk. Since DOS isn't multi-tasking, personal phone access is readily done the normal way. And all editors offer "macro" capability in some form or other.

In fact, if your editor allows you to exit to DOS by loading a secondary command interpreter (as most do), you can run everything from your editor, using it as a substitute operating system, as it were, which is what I do most of the time.

*Morton E. Kaplon*
*Easton, MD 21601*

**InfoConnection,**
**More Undocumented DOS Functions**

Today I am mailing my check for a one-year subscription: I've purchased the last several issues at Silicon Valley's cultural center (Computer Literacy Bookshop) but was unaware until you pointed it out that you aren't really making money on newsstand sales.

For many years I have subscribed to *InfoWorld*. However, this timely publication has apparently shifted its audience target from developers to users, from people in the industry to consumers of the industry's outputs. For example, *InfoWorld* will warn (users) of a potential incompatibility problem with the 80386 chip but never actually tell (developers) what the incompatibility is. Consequently, I am looking around again for news and views from the inside. Perhaps *Programmer's Journal* will develop into that InfoConnection.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

I was surprised that M. Steven Baker's "Terminate and Stay Resident" in the November/December 1986, issue did not mention these undocumented DOS functions:

AH = 51h, output BX = current active task id
AH = 50h, input BX = new active task id (from function 51h)

Each program segment prefix contains a table that translates file handles into indexes into a master table inside MS DOS; 51h and 50h control which translation table is used and hence whose set of file handles is actually live at the moment. In order to perform I/O on your own file handles, you must:

upon startup
(0) get active task id as "mine"
at take-over time:
(1) get active task id, save as "his"
(2) set active task id to "mine"
(3) do your I/O
(4) set active task id to "his"

This does not solve the critical error handler problem, which is a mess in itself. Functions 51h and 50h are present in this form in PC DOS 2.0, 2.1, 3.0, and 3.1 (I haven't checked 3.2). I also suspect that it controls the ownership of any memory blocks allocated via function 48h. At present the "active task id" is always the segment address (implied 0 offset) of the program segment prefix; that is likely to be a numerical convenience that could change at any time.

Thanks for your efforts to create a good solid magazine for us techies; I'll look forward to reading it in the coming year.

*Andy Canfield*
*Sunnyvale, CA 94089-2328*

**Furthermore. . .**

Finally, this morning (Sunday) I had a chance to relax. One of the things I did was to read the November/December 1986, issue of *PJ* cover-to-cover. The stuff that's there is pretty good—of course there could be more (no one's ever satisfied). Reading the two DOS articles confirmed my happiness at not having to develop DOS applications. The few that I have done led me into those murky waters M. Steven Baker talked about, and what with insufficient tools, I was quite turned off to DOS.

Some of Mr. Baker's comments have also prompted me to collect my thoughts vis-a-vis the difficulty of creating a multi-tasking DOS that is compatible with existing DOSes. Recently, I was asked by AT&T's Industrial Automation Group for a technical analysis of AT&T's PC-6300+. Here's a 286 machine with about $50's worth (cost) of hardware, which, combined with Locus Computing's software, allows an amazing number of (often ill behaved) DOS programs to run as tasks under UNIX. Bits of circuitry trap writes to the DMA controller, for example. The register loads are examined to see if the results will write over UNIX or something else that shouldn't be written over, and so on. The AT&T people wanted to know, "Why won't it run *all* DOS programs?" The answer was by then obvious: Without an even more extensive effort than Locus's OS-MERGE (a.k.a. Simultask) and more specialized hardware, it is impossible to have a multi-tasking DOS that will be fully compatible. No wonder Microsoft and IBM are loath to release such a DOS. You can just hear the bloody screams now!

*Gregg Squires*
*Mnematics*

**CTOS**

I just read Frank D. Greco's "286 DOS Cometh" in the January/February 1987, issue. Systems programming is a bit outside my field, but I found it interesting to compare Microsoft products as described in the article with another 8086 operating system. I worked for a couple years on a system of desktop 8086 machines made by Convergent Technologies (also sold by Convergent OEMs such as Unisys). CTOS (Convergent Technologies Operating System) is multi-tasking, multi-user, and richly endowed with services for the OEM, supplying system extensions depending on low level hardware—and has been that way for five years. No direct access to the hardware is ever used, however, because of the richness of the system services. And the system and all applications generated in high level languages are reentrant. The CT desktops have built-in network communications hardware, and the network is built into the operating system.

Interrupt driven, buffered I/O in CTOS is a system service; it creates, maintains, and operates buffering for all devices. You never write device drivers—they are all supplied. The keyboard driver, for instance, maintains two buffers: one for regular keys and another for "action" keys (engaged by a third "action" Shift key) that are used to interrupt processes. Yet interrupts for additional devices and services are provided to integrate anything you want into the multi-tasking system as part of the system, not the application. The system itself is supplied as an object code library and macro files that you can link to generate systems with different configurations.

To use system services, you can either call operating system procedures (provided in object code libraries) or interface directly to the multi-tasking system through semaphores.

My point is that the limitations of programming in the Microsoft operating system are not in any way due to inherent requirements for working in a multi-tasking environment—CTOS doesn't restrict the capability of applications. The greatest difficulty in programming IBM computers stems from the failure of the operating system to contain device drivers.

The only problem with CTOS was that the hardware was relatively expensive and not enough people supplied applications for it.

*George Lyons*
*Jersey City, NJ 07302*

**More on Inline Code, Word Perfect Defended**

I would like to make some comments on James Smith's article about Turbo Pascal Inline Code (*PJ*, 4.6).

I hope that Mr. Smith is as happy with his "forget portability" attitude in five or ten years as he is today.

It is worth noting that some of Mr. Smith's proposed conversions are of a nature that could affect compatibility with future versions of the Turbo Pascal compiler. I would be very careful about changing procedure entry and exit mechanisms if being able to use Turbo Pascal Version 4 is an issue. Since Version 4 is supposed to be able to generate linkable .OBJ files and programs larger than 64K (.EXE files), I am hesitant to endanger compatibility with it.

I do agree with Mr. Smith that there are some cases where reduction to assembly language is worth the lost portability.

By segmenting out the few routines that you want to convert to assembly language, you accomplish a number of things:

1. You keep the assembly code small and manageable. Large pieces of assembly code are *much* more difficult to maintain than small ones.
2. Most of your code remains in the higher level language, making it easier to port to another system.
3. Small assembly routines can be well defined and well documented. This makes rewriting them for another system much easier.

One of the big advantages of using inline code, and one I am surprised Mr. Smith overlooked, is the size reduction for programs that use overlays. If a program is not far over the 64K limit, re-coding into inline assembly language can frequently eliminate the need for overlays. I consider this the best reason for using inline code in more than a few critical routines.

Incidentally, a public domain Turbo Pascal program called INLINE.PAS takes as input an assembly language file (for the 8086) in a format similar (but not identical) to that used by the Microsoft assembler and generates an inline statement for that code. It's a pretty handy program, and one I use for generating inline code.

Mr. Smith seems to be under a common


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# 2 Programmers & Developers New Products!

## Distribute Your Demos with No Royalties

**Screen Machine** creates interactive demos, tutorials, menu systems and DOS shells. Includes a text screen editor that optionally generates source code* and binary or text files. Never write code for screen display again. Capture any program's text screens for editing and your own use. Capture CGA compatible graphics screens for BLOAD or direct display. SAVE hundreds of HOURS of work.

Now there's no need for separate screen and demo software packages and no need to pay outrageous royalties. Priced at only $79.00.

*Turbo Pascal, Mach 2 for Turbo, Assembler, dBASE II & III, BASIC (including The Inside Track and Mach 2).

## Supercharge Turbo Pascal

**Mach 2 for Turbo Pascal** adds assembler speed to your programs. 90 + subroutines, most in assembler, give you speed and functionality you never knew was possible. No knowledge of assembler language required.

INSTANT displays. INSTANT windows (incl. exploding and boxed). FASTEST sort you've seen. Read/write files FAST as DOS. INSTANT menus, 1-2-3 horizontal and vertical bar.

Trap ^C/^Break & DOS critical errors so no more A)bort, R)etry or I)gnore. Emulate BASIC PRINT USING for FAST formatted numbers. Execute any prog, batch or DOS command without ending program.

Read environment. Read file directory. Get/set file attributes. Plus too many string functions to describe here. No royalties when you distribute COM programs. All source code included. A true bargain at $69.00.

---

NOT COPY PROTECTED. 30 Day Money-Back Performance Guarantee. Requires IBM/compatible & DOS 2+.

## Order Now 800-922-3383

We welcome VISA/MC. COD US only $3. S/H US $3, Canada $5, Elsewhere $18. GA res. add tax and call 404-973-9272. Demo available. Send $5 check. Refunded on direct purchase.

We also publish Stay-Res, Mach 2 for BASIC, The Inside Track and Peeks 'n Pokes.

**MicroHelp, Inc.**
2220 Carlyle Drive
Marietta GA 30062

Number 90 on the Reader Service Card


misconception concerning 8086 assembly language. A conditional jump is *always* a short jump. In a statement such as

```
JZ  SHORT   out
```

the "SHORT" is superfluous. Only the JMP instruction necessitates a range qualifier.

While I would question in general what 80% of Hal Hardenbergh's November/December column (*PJ*, 4.6) is doing in a "Resource Journal for IBM PC Programmers," I take particular exception to his comments concerning Word Perfect. More than just the factual errors or the negative aspersions cast on an excellent product, the total closed-mindedness on the subject really frosts me.

The first thing Hardenbergh complained about is having to re-enter all his format codes. Word Perfect is capable of importing and exporting document files in the following formats while preserving most, if not all, format codes:

Revisable Form Text
Final Form Text
Navy DIF Standard
WordStar
Multimate

If Spellbinder is not capable of outputting files to one of these formats, I would say that was a shortcoming of Spellbinder, not Word Perfect.

*Brad Chase*
*Exeter, NH 03833*

### Use the Alt Key

A slight correction to the correction in your January/February 1987, issue (ERRATA, Turbo Inline Code): You can use invalid characters (specifically the IBM extended character set) to make an "invalid" filename in DOS. The trick is to type in the ASCII number using the Alt key. For example, to put a space in a filename, hold down the Alt key and type 255. To get into the file, just enter the name the same way. The charactrer can be at the beginning, middle, or end, and the trick works for subdirectory names, too. For a mild form of snoop-proofing, I made a subdirectory whose name was nothing but Alt-255 Alt-255 (that is, two spaces). The subdirectory shows up in a directory listing as a blank line followed by <DIR>, and you can't get in without knowing the trick although there are ways to get a listing of the filenames therein.

Incidentally, Microsoft Word uses this same Alt-key technique to enter extended characters (for foreign alphabets) into a text document. Here I found a problem—and a semi-kludge solution. The problem is that some printers won't print the IBM extended set. So I use Alt-136 (which the printer reads as a backspace) to combine an "e" and an apostrophe, for example, to get an accented "é." I don't enter Alt-8, by the way, because it doesn't put a backspace character in the text; it just *does* a backspace and wipes out the character before. As Arte Johnson used to say, "Verrrry interesting . . ." Anyhow, there are lots of combinations: a "Y" plus "=" produces a Yen sign (¥); quote marks (") can fake an umlaut (ö), etc. Try out a few.

Love the magazine, especially DTACK GROUNDED!

*Fred Chapin*
*New York, NY 10014*

### More on Calling DOS from Turbo Pascal

I would like to congratulate Mr. James M. Smith for his excellent article on Turbo Pascal inline code in the November/December 1986, issue.

On the other hand, Stephen K. O'Brien and Wing Kit Cheung mention that they "don't find using inline code very instructive."

They detected an error in Arvind Kumar's "Using DOS to Call Programs from Turbo Pascal" in the January/February 1986, issue of *PJ* and proposed an external procedure to call DOS's function 4Bh as a solution.

Unfortunately, if you disassemble O'Brien/Cheung's solution, you'll see that the offsets to the code segment variables (SSSAVE, SPSAVE and BPSAVE) they used in the routine point to the beginning of the Turbo program code, not to the beginning of the external procedure's code. MASM does not produce relocatable code for external program code segment variables such as those used by O'Brien/Cheung, and neither does Turbo take account of them at compile time. (They could have saved the instruction pointer first and then added that value to the variables' offsets.)

I'm not sure that the Fork and System functions are working properly, with either Kumar's or O'Brien's code. I tested them "forking" Btrieve version 4.01 by Softcraft (the resident file handler), and they seemed to work error-free, but CHKDSK would then show approximately 30K less memory available than when I had run Btrieve from the DOS prompt. (When I ask Btrieve to unload itself, it does free all the allocated memory.)

Can anyone tell me how to execute a memory resident program from Turbo without trapping extra memory?

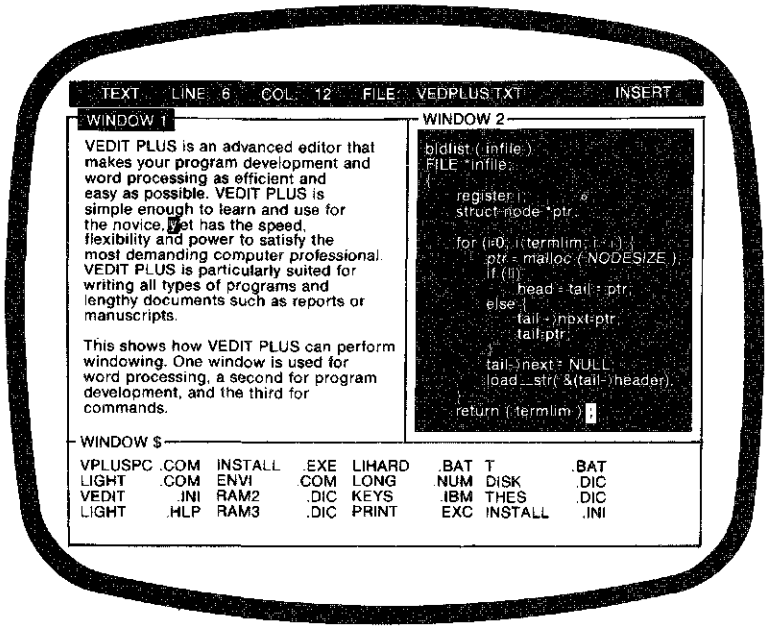
*Garza Garcia*
*NL, Mexico 66220*

*If you invoke Btrieve from the DOS prompt, it is loaded on top of DOS. If you fork it from your program, it is loaded on top of your program. When your program terminates, it leaves a "hole" in memory that DOS can't recover, and Btrieve remains resident above the hole. The missing 30K is most likely the size of the parent program.*

*Considering the variety of ways in which TSRs can behave, invoking a TSR from a program may give unpredictable results. Some TSRs may make the assumption that they are installed beneath application software as part of their method of deciding when it is safe to operate. Such a TSR may fail to operate at all if invoked atop a parent. —Ed.* □



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED


TEXT LINE 6 COL 12 FILE VEDPLUS.TXT INSERT
WINDOW 1
VEDIT PLUS is an advanced editor that makes your program development and word processing as efficient and easy as possible. VEDIT PLUS is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. VEDIT PLUS is particularly suited for writing all types of programs and lengthy documents such as reports or manuscripts.
This shows how VEDIT PLUS can perform windowing. One window is used for word processing, a second for program development, and the third for commands.
WINDOW 2
WINDOW $
VPLUSPC .COM INSTALL .EXE LIHARD .BAT T .BAT
LIGHT .COM ENVI .COM LONG .NUM DISK .DIC
VEDIT .INI RAM2 .DIC KEYS .IBM THES .DIC
LIGHT .HLP RAM3 .DIC PRINT EXC INSTALL .INI

# VEDIT PLUS

## #1 CHOICE IN PROGRAMMABLE EDITORS

VEDIT PLUS has been the #1 choice of professionals since 1980. Our latest release is even better - you can open windows to simultaneously edit several files, access many editing functions with pop-up menus, use keystroke macros to speed editing, and run other programs or DOS commands from within VEDIT PLUS.

Whether your needs are program development, technical writing or word processing, VEDIT PLUS is your answer. With over 40,000 users you can depend on VEDIT PLUS to perform consistently and reliably. It is simple enough to learn for the novice, yet has the speed, flexibility and power to satisfy the most demanding professional.

Compare. No other editor is as powerful - unlimited keystroke macros, instant 'off-the-cuff' command macros utilizing a complete programming language, single command file comparison, special word processing and programming features. No other editor is as easy to use - on-line help, pop-up menus, 75 page tutorial, 380 page manual, and VEDIT PLUS is completely customizable.

Fully supports color windows on IBM CGA & EGA, and even windows on most CRT terminals (including CRT's connected to an IBM PC). Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. Order direct or from your dealer. $185

"To sum things up, VEDIT PLUS is a small, fast, sophisticated editor with a wealth of features and a good macro language. It offers many rewards for the dedicated programmer."
**Computer Language, Chris Wolf, Scott Lewis, Mark Gayman 6/86**

"VEDIT PLUS is a wholly remarkable program: blindingly fast, extremely powerful, and highly flexible."
**Profiles Magazine, Robert Lavenda 4/86**

### VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname and CP/M user number support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 text registers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size, any CRT.
- Optimized for IBM PC/XT/AT. Also 132 column & up to 70 lines.

**EASY TO USE**

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

**FOR PROGRAMMERS**

- Automatic Indent/Undent for 'C', PL/I or PASCAL.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

**FOR WRITERS**

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert WordStar and mainframe files.
- Print any portion of file; separate printer margins.

**MACRO PROGRAMMING LANGUAGE**

- 'If-then-else', looping, testing, branching, user prompts keyboard input, 17 bit algebraic expressions, variables.
- CRT emulation within windows, Forms entry.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Main menu

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research. WordStar is a registered trademark of MicroPro.

**CompuView**

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

Number 56 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# DTACK GROUNDED, The Junk Mail Flyer

by
**Hal W. Hardenbergh**

*In this most recent communication from Hal Hardenbergh, we jump into his marketing plans for DBASIC, Hal's BASIC interpreter-cum-compiler for the Atari ST.*

A logical marketing plan would be to put a price tag of $129 on DBASIC, copy protect the disk, and advertise as heavily as our budget would permit. I honestly believe that DBASIC could sell to about 2% to 3% of the market on that basis, which would mean gross sales of $1 to $1.5 million in calendar '87. That's about twice what Digital Acoustics has sold per year recently. That would be quite profitable, even after subtracting fixed costs, overhead, and advertising.

Well, we *have* been pouring money into DBASIC since 1981 if we count the 68000 floating point package, or since 1982 if we count the start of HALGOL. We are capitalists, and we believe in profits. But we're *not* going to follow that conventional marketing plan.

**Manual Printing Costs:**

Some background. James has been investigating printing costs and methods. The most cost-effective way to print a manual uses what is called "perfect binding." Look at a Borland manual, for instance. A 250-page manual sized 8.4" by 5.4" costs about $9,100 in quantities of 7,000, or about $1.30 each. Add a dime or so for cover artwork, and you are looking at $1.40 per manual.

Such a manual weighs about 10 ounces, so 7,000 of them is about 4,400 pounds or over two tons of manuals. The manuals are delivered in cartons and/or on pallets. If they were all on one pallet, 7,000 manuals on a 3 foot by 4 foot pallet would be a stack ten feet high! 7,000 manuals will be our initial buy. If you get the impression that we are not playing with DBASIC as a hobby, you are correct.

The reason manuals with "perfect binding" are so cheap is that the process is *completely automated*. An alternate approach is the three-hole binder that is typical of, for instance, IBM's BASICA manual. Usually the three-hole binder is provided separately, with the manual pages enclosed in shrink-wrap. Here is how these are made:

To get shrink-wrapped three-hole manual pages, the printer prints up the usual "perfect binding" manual, with a blank cover. The binding holds the manual pages in place while the three holes are drilled. Then the binding is sliced or sawed off, and the pages are shrink-wrapped. This last is a manual operation, and so the manual costs a lot more. Say, $1 to $1.25 more per manual.

The binders, including the ones that seem to have cloth covers, have vinyl covers. Cloth may be impregnated into the vinyl. The covers are usually silk-screened with a title and logo. Then the vinyl is manually folded and tucked around the cardboard to make the three-hole binder. Because this is all *manual labor*, the binders cost nearly $3 in quantities of 5,000. What's worse, the delivery time is six weeks at best and more typically eight to ten weeks. And it can be worse.

Delivery of "perfect binding" manuals is usually quicker because the process is automated and the only manual labor is packaging the manuals for delivery.

Believe it or not, the fact that the usual three-ring binder and manual costs about $5.50 vs. $1.40 for perfect binding is not all that terribly important. The typical delivery of three weeks vs. ten weeks is *vitally important at this time*!

**Ya Gotta Stock Up:**

You see, we have to place our initial order for manuals *before* we have begun to sell DBASIC, and therefore before we know how well DBASIC will sell. Since 30 days is the absolute outer limit that can be tolerated when ordering software (and there are even some Federal rules on that), we *must* have on hand enough manuals to handle the *upper limit* of possible software sales. (When JRT offered that $29.95 PASCAL a few years back, this was one of the problems that caused JRT to go belly up.)

Suppose that we place an absolute upper limit on possible DBASIC sales of 5,000 per month. That means we have to initially stock about 5,000 manuals if we use "perfect binding" or about 12,500 manuals if we use 3-hole binders. 5,000 times $1.40 is $7,000. 12,500 times $5.50 is $68,500. Now you know why the delivery time is so vitally important, and why we are *not* going to use 3-hole binders.

Do we actually *expect* to sell 5,000 copies of DBASIC per month? Of course not! But we are (going to be) *prepared* to sell that many, from the standpoint of personnel, cash flow, and stock on hand. (Poor JRT!)

How many do we have to sell to make a living? Our overhead is about $20,000 per month. If we sell the software package for $50 and have a gross profit margin of $40 per package, we have to sell 500 copies of DBASIC per month to break even.

That's 6000 sales in one year to break even, against an English-speaking installed base which is estimated to be 375,000 at the end of that year. That's 1.6% of the installed base. I think the world's fastest interactive BASIC (sans math chip division), priced at $50, can sell that well. If I'm wrong, I can always get a job somewhere else. (One fun thing about running a small business is getting to play "*you bet your company and your livelihood*" frequently.)



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

So we are geared up for sales of between zero and 5,000 on the first month in which DBASIC will be offered for sale. We have the financial resources to ramp shipments up to 10,000 per month within three months, in the unlikely event that that should prove necessary.

**Raucous Laughter:**
We sent a first draft of this writeup on our marketing approach to a regular reader whose company sells some software into the ST market—an application priced at about $150. He made numerous photocopies of it and handed them out, which occasioned much hilarity, mostly aimed at the numbers you have already read. You see, the first printing of *their* ST manual was fifty (50) copies, which sold out. The second printing was for two hundred (200) copies, which is about to sell out. There is vigorous discussion, both pro and con, within the confines of that company over whether they should risk printing an entire one thousand (1000) manuals! Meanwhile, Gfa BASIC for the ST goes on sale in West Germany and moves 5,000 copies in the first 4 weeks. . .

A 99 foot ABEKING RASMUSSEN twin-diesel yacht goes for $1.9 million these days. Add 6% sales tax and a little negotiating and that's $2 million even. That's a lot of DBASIC sales, especially after Uncle Sam takes his cut. Hmm. Did I tell you about the profits we are going to make selling our checkbook balancing program that will run on the ST using Basic09 and OS-9? Oh, yes: the DBASIC source code now weighs about 5⅓ lbs.

**Our Marketing Plan:**
As I have already explained, our objective is not to get top dollar from 3% of the installed base. That would be a shortsighted objective. If you have the best (i.e., fastest) available BASIC for the ST, as I believe we have, then that BASIC should become the *standard* BASIC for the *entire* installed base, at least the English-speaking portion. When I say entire, I mean as in 100%! That's because some demo, game, and application programs written in DBASIC will be used by even the drooling rock-shooters who will never write a single line of program code.

**One fun thing about running a small business is getting to play "you bet your company and your livelihood" frequently.**

*Let's devise a method of getting DBASIC to 100% of the installed base of ST owners*: First, we need to get DBASIC to the 40% (my estimate) of ST owners who program in BASIC or who want to learn how to program in BASIC.

Run a massive advertising campaign? C'mon guys, be serious!

Each of the 300 ST user groups can expect to receive two manuals, two disks, and about 100 labels. That's about 30,000 labels right there. Another 50,000 labels will go out with the first 5,000 purchases alone. *Gee, I sure hope nobody pirates our software*!

We have told you in the past that we cannot sell outside the U.S. That's only partially true. We can't sell the software in machine-readable form (diskette) without an export license. But *there are no such restrictions on printed material*! That means we can sell manuals and diskette labels complete with Digital Acoustics copyright notices, and printed authorizations to copy a DBASIC diskette, provided one of our labels, the one with the copyright notice, is attached. Since we cannot ship a diskette to a foreign DBASIC buyer, we can reasonably ship about 100 such labels instead.

Where would a foreigner find a DBASIC diskette to (legally) copy? *Beats me, Jack*! But I suspect one could be found, especially if *one* such disk clones to *100* absolutely legal copies! And several of the recipients of those absolutely legal copies might want to buy manuals from us. Each manual would arrive with an additional 100 labels bearing copyright notices, plus authorization to use those labels.

It costs *money* to make 100 copies. Are those foreigners going to *sell* the copies? We could care less. (An unlovely phrase, but one which accurately reflects our sentiments.) After all, they *bought* those labels, ownership has been transferred, and *we do not wish to tell others what to do with their property*!


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Of course, it would be absolutely illegal, under international copyright conventions, for those foreigners to make more than 100 copies. And so they are directed *not to do so*! We certainly don't want our software to be pirated!

Why not sell DBASIC for less than $50 if we want it to be universally used? Because *handling an order is a manual operation*! My estimate is that a total of nine person-minutes, altogether, are required for each order. Unlike Philippe Kahn, I am not a people person, so I want to run a small, tight operation.

That means we have a conflict. I want every ST to run DBASIC, but we can't handle that many individual sales transactions. So we sell through a distributor? *Wrong*! I would sooner jump into a pit filled with rattlesnakes.

You don't have to have a DBASIC manual to run programs written in DBASIC. All you need is a diskette with a proper label printed by us with our copyright notice on it. For those of you who suggest that a pirated copy would serve as well: *Shame*! Go stand in the corner! Nobody is going to pirate DBASIC; after all, software piracy is *illegal*!

One way to hold down people costs (sales transactions) is to include two manuals in each $50 package! Here in the U.S. we can include a master diskette, so we will toss in only ten or so labels plus written authorization to use them. So the person who has an ST at home and another at work can have a manual in each place. And two students can get a manual each for $25 each *without our having to handle two sales transactions*. The extra $1.40 is insignificant against the $50 purchase price, and postage is $2.40 for parcel post up to two pounds, so that extra $1.40 is the only extra cost involved in shipping two manuals!

We are prepared to handle as many as 5,000 sales transactions in the *first* 30 days, which means 10,000 *manuals* per month beginning in the *first* month! We will order 7,000 manuals initially and give away about 2,000 of them in the process of "seeding the market." So our stock of 5,000 manuals would last 2 weeks at the maximum order rate. Since we can get more in three weeks, that means that we would have no more than a one week delay in getting more manuals. (Naturally, sales will never ramp up that quickly. And we are also prepared for *zero* sales the first few months.)

**Minimizing Sales???**

What's that? What if the sales are *more* than 5,000 per month? Can't happen. We will have control of the telephone answering service. After 232 orders are received on a given day, *we shut down the phone*! (There are about 21.5 working days in a month.) No, I am not joking.

The reason I have given such emphasis to the *maximum* order rate, and how to prepare for it, is that a company can just as easily go out of business because of *too many* orders as because of too few orders. If you don't believe me, ask James R. Tyson, who is now flogging Mystic Valley PASCAL.

Personally, I think sales of DBASIC for the first few months after it goes on sale will be somewhere between zero and 5,000 per month. (Talk about making a fearless prediction!)

**HP Watch:**

In the 25 Mar'86 *JMF*, on p.7, I wrote: ". . .there has been a shortfall in the actual as opposed to theoretical performance of Hewpie's new RISC machines. A performance increase of 5 had been expected with respect to the old 3000 series. Rumor says the actual measured performance increase is between 2 and 2.5 to 1."

## "This is a hell of a time to figure that out."

I received several letters as a result of that paragraph. The letters pointed out how competent HP was, how long the RISC machines had been in development, how many computerized studies must have been made of its performance, etc, etc.

Ahem! *Electronic News*, Oct 6, p.16:

"Since disclosing additional delays in bringing out a general-purpose system based on its Spectrum RISC-like architecture, Hewlett-Packard also is *admitting earlier throughput claims were never actually met*. When HP introduced the Series 930 earlier this year, Joel S. Birnbaum, vice-president and director of HP Labs, said that based on tests conducted by the company, 'we find it delivers at least twice the throughput' as the series 68. An HP spokesman now maintains Mr. Birnbaum meant to say the 'goal' was for the Series 930 to deliver twice the throughput and that *the 930 was not achieving such a throughput level at the time and still isn't*. The spokesman said the firm still expects to achieve greater than double the throughput levels after additional software tuning, but declined to disclose what the machines are currently attaining [emphasis added]."

In that same *EN* issue, on p.21, we discover, ". . . software problems with the portion of its operating system controlling I/O. That difficulty is hindering system throughput and causing performance shortfalls. . . HP's explanation that it had just discovered the I/O problem was questioned. 'This is a hell of a time to figure that out,' Mr. Crawford said. . ."

Could it be that there really is a connection between the performance of the RISC and Hewpie management's sudden, panicky preemptive anti-hostile-takeover move in 1985?

**68030 Report:**

Motorola has released details of its next microprocessor, the 68030. The good news is that it will, in fact, be about twice as fast as a 68020. The bad news is that it is very, very late. It will be a *long* time before you will be able to buy a personal computer featuring a 68030 CPU.

Motorola has also announced that a second-generation math chip with twice the speed of the 68881 will be delivered with the 030. Uh uh. The past several years have proved that Motorola has enough skilled hands to work on only *one* VLSI project at a time. The 68881 was terribly late arriving because the 68020 and its memory management unit both had priority. The *next* Motorola math chip will be terribly late arriving because the 68030 and 78000 microprocessors will both have priority, along with maybe another chip or two. Remember, the 68881 was promised for Feb '82 and arrived as first silicon in Aug '85.

The 78000? That's a 32-bit RISC-like chip which may arrive before or after the 68030; Motorola is playing it cagy. Now that the shortfall in performance of chips designed to the pure RISC dialectic has become evident, nobody makes RISC chips any more, just RISC-like chips.

Who is going to be the first bozo to assert that the 68030 is so fast that the operating system can be written in Pascal without anybody's noticing? (I have already lost track of the number of bozos who have asserted that the 386 is so fast that everybody is going to bit-map and window all over the place. Sure. And judge Crater is going to show up any day now.)

**Theory vs. Practice:**

Both the Nat Semi chips and the AT&T 32-bit micros were supposedly designed to be optimum for C. And yet every time third-party benchmarks written in C are run, those two micros consistently drag across the finish line in a dead heat for last. See



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

the comparisons in Aug *IEEE MICRO*, for instance. (Incidentally, those *IEEE MICRO* comparisons confirm my assertion that the 68020 will run off and hide from the Intel 386 if the 386 has no external cache.) Well, the Intel 432 was supposedly optimized for Ada.

There is a fallacy here someplace. Microprocessors written specifically to run C do so more slowly—*much* more slowly—than more general purpose architectures such as the 680X0 series. The cupboard of the computer scientists and their sacerdotes is bare; they have no explanation why this is so. *Since the facts do not conform to theory, the computer scientists simply ignore the facts*!

What other microprocessors were designed specifically to run C? Why, just about all of the RISC machines! For you new arrivals, a RISC machine is something that a computer scientist asserts will have 150% of the bus traffic of a VAX for equivalent software but that turns out to have 450% of the bus traffic of a VAX when a third party benchmarks it. Or, a RISC machine is one which, before it is introduced, is going to be five times faster than the old machines but which runs slower and slower as introduction approaches. (Has anybody noticed that HP's RISC is now less than twice as fast as the old HP machines even though *it has not yet been released to the outside world?* How much slower than *that* is it going to be when—if—it is finally released?)

RISC machines used to be much faster when all you could find out about them was what a computer scientist wrote about them in a refereed journal.

**Qualified vs. Competent:**

A recent column by Jerry Pournelle raised once again the old issue of qualifications vs. competence. As the old joke goes, what do you call the guy who graduates last in his class at medical school? Why, *Doctor*, of course!

A great deal of the nonsense floating around in the computer field comes (as Jerry suggested) from the academic world where folks worry about what academic degree one has and at what institution it was awarded.

**17 Miles of Landfill?**

The *WSJ* has reported that sales of the UNIX PC are "under 2000" in calendar 1986 to date. That means that not even AT&T will buy that turkey any more. Fortune continues to make progress towards spending all of the $110 million it started with. Less than two years ago, the president and the marketing manager of a small 68000-based microcomputer company which did not then build UNIX boxes told me they were going to introduce UNIX on their machines. I commented that that was what a lot of companies did *just before they went out of business*. One year later that president and that marketing manager were no longer with that company. Surprise?

I find myself at a mixed-media happening. There are spotlights, a laser show, and the Goodrich blimp circling overhead. In stage center, AT&T CEO Olsen announces the retirement of AT&T from the personal computer market, AT&T having managed to drop *ONE BILLION DOLLARS* proving that it knows as much about that market as about concrete sewer pipe or women's brassieres. On stage left is an announcement of the absolutely last, final UNIX standard of this week. On stage right a Greek chorus is bemoaning something or other; too bad I don't understand Greek.

In the center of the arena, showing up nicely against a background of bulldozers, is Jean Yates. Jean is about to turn the first symbolic spadeful as 17 miles of UNIX PCs are about to achieve their ultimate destiny as landfill. As Jean lifts the shovel, the Sun Micro solid-gold corporate Lear Jet passes overhead. Under one wing is the inscription, "ABSOLUTELY, POSITIVELY, OFFICIALLY STANDARD UNIX." Under the other wing is "ONE MAN, AT LEAST ONE 25MHz 68020." Dreaming? Who, me?

AT&T has achieved the dubious distinction of being the first large American cor-

SPEEDWARE™
A DIVISION OF AMERICAN AFFORDABLE SOFTWARE CORPORATION
PRESENTS
Turbo Debugger™ Features
• User-Definable custom windows — Change window sizes or colors
• Single-step tracing with "Back-Tracing" up to 100 instructions
• In-Line Relocating Assembler with non-destructive insert capability
• Fully Symbolic. Create Comments and Line Labels. Keyboard macros
• Dual Monitor support, with Screensave
• Specify several Break-Point conditions with the built-in Assembler
• Use NMI Breakout switch or User-definable hotkeys
• Read, Write and Modify .EXE files
Turbo Editasm™ Features
• Integrated WordStar compatible Editor
• Integrated MicroSoft compatible Macro Assembler
• Full 8088, 8086, 8087, 80186, 80287, 80286(R) & (P) support
• Full Macro expansion capability
• Built-in Calculator & Xreference Capability
• Limited DOS command support
• Supports In-Memory Assembly
To maximize your programming potential order the ultimate "Turbo Team" today — the first well-tuned, high-speed assembler/debugger for the IBM P.C. and compatibles family.
$99.00 Turbo Editasm™
$89.00 Turbo Debugger™
THE TURBO TEAM™
Inside California (800) 334-3664 • Outside California (800) 443-6190
$99.00 ☐ Turbo Editasm™
$89.00 ☐ Turbo Debugger™
Name
Shipping Address
City
State Zip
Telephone
Amount Enclosed
Payment VISA MC BANK DRAFT CHECK
Credit Card Exp Date
Card #
CA Residents add 6% sales tax.
Outside USA add 10% and make payment by bank draft payable in U.S. dollars drawn on a U.S. bank.
Dealer and Education Discounts Available.
Speedware™
9719 Lincoln Village Dr., Ste. 303
Sacramento, CA 95827




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

poration to drop a billion dollars in the personal computer marketplace, *without Kindly Uncle Jack being P.O.d at it!* (Remember T.I.?)

**Transputer Revisited:**
One, and only one, of the following statements is true: (A) No computer architecture will ever prove superior to the von Neumann architecture for *general purpose* computing. (B) A computer architecture will eventually be devised which is superior to the von Neumann architecture for *general purpose* computing.

I have no idea which of the above statements is true. I do know that a lot of candidates to supplant the von Neumann architecture have entered the lists, only to retire in disarray. *It does not seem to be a smart move to gamble real money on any one specific candidate.* It does not surprise me that the British government provided the initial funding to back Inmos and the Transputer because governments often do stupid things. But Floating Point Systems has had a *long* history of success and sensibility!

As far as I know, Floating Point Systems is the only sizeable computer manufacturer that has announced a *general purpose* computer based on the Transputer, the Series T. Before beginning development of the Series T *FPS had been profitable for 28 consecutive fiscal quarters.*

FPS dropped $2.95 million in the second calendar quarter of 1986 (the *operating* loss was $3.9 million) and is expected *not to do that well* when the third quarter results are announced.

**More Transputer:**
"FPS' chief scientist for the T Series, John Gustafson, [has departed]. He said that he had become dissatisfied with the progress of the project, and cited problems with the INMOS Transputer used in the new supercomputers.

"The [Transputer's] high speed serial ports. . . proved to be inadequate for the needs of the hypercube arrangement, according to Gustafson.

"The reduced instruction set architecture of the microprocessor also turned out to be a further limitation. 'We found that it took a minimum of 40 instructions to do the most minimal operations needed,' [Gustafson] said." (*EET*, 10 Nov, p4)

**Amateur vs. Professional:**
Those of you who are giggling over our amateurish marketing plan for DBASIC will undoubtedly appreciate the more professional approach by Microware for its OS-9 operating system and Basic09. The licensing fee is $300 for the operating system and another $300 for the BASIC. Let's see now: Carry the five. . . Hey! That's *$600*! And they don't throw in a hundred or even ten more authorized, copyrighted labels, and I'm almost sure you don't get two manuals.

But they are going to make fifteen times the gross profit per sale for their OS and BASIC than we will for ours! Gee, why couldn't *we* have been smart enough to think of that?

We will make up for our oversight by raking in enormous profits with our $300 checkbook balancing program for the ST, which will be written in Basic09 and run on OS-9. And we will need the money, 100 foot twin-diesel yachts being priced higher than I had anticipated. $1.9 million for a rather elderly 99 footer??

**386 Expert Idiots:**
Intel has finally come up with a chip called the 386 that can linearly address more than 64K. Naturally, there are a lot of expert idiots who are predicting that UNIX will be the standard operating system on mass-market 386 machines! I am sure that Jean Yates will concur ten million per cent.

**The cupboard of the computer scientists and their sacerdotes is bare.**

There is more: The Intel folk are falling into the identical trap that the 680X0 folk fell into years ago; now that they have a high-performance part that is not memory-limited, they are going to have bit-mapped icons and windows. They are going to have virtual memory. They are going to write their operating system in a high level language. They are going to have *multieverything*. And above all, UNIX is going to be their operating system! Terrific.

**Rationality Surfaces:**
Besides the 20% of the Intel folk who have learned the facts of life, there are even some 680X0 folk who are wising up. *Computer Shopper* carries a curiously-titled article comparing the Amiga and the ST and concludes, "I hate GEM! I hate GEM! Did I tell you I hate GEM?" Issue #2 of the Boston Computer Society's *JBUG*, which covers the "Jackintosh," carries a commentary by Dick Miller of Miller Microcomputer Services which correctly points out that GEM is a disaster (performancewise) and that TOS seriously needs fixing.

Yes, there is now a significant minority of folk involved professionally in the personal computer game who know that "*the chip is so fast that you won't notice*" is for expert idiots, losers who want to produce the next LISA. Read, for instance, Bill Machrone's editorial entitled "The Death of Good Software" in 25 Nov *PC*.

(Bill rhetorically asks why MicroPro's *Easy* word processor ". . . is such a pig?" Simple, Bill: A committee of 50 Modula-2 coders wrote Easy—in wonderful Modula-2—in nine months.)

Question: When I read a review such as the one on the Mac Plus in Nov *BYTE* and see that when the reviewer tried to save an 8K file from a megabyte-RAM machine onto an 800K floppy, *he had to give up and hit the reset switch after 13 disk swaps*, should I laugh or cry?

Every now and then I put a clipping away to ripen. I have beside me a guest editorial from *PC Magazine*, 11 Dec '84. It says that UNIX is about to become (as of Dec '84) the standard operating system for the AT (and the 286 processor). Today one reads the identical material except that the CPU in question is the 386! Well, the 16-bit micros are history, suitable only for rock shooting toys (68000 division) or single-tasking business purposes (286 division). If UNIX does not manage to become the standard operating system on the 32-bit generation (and it won't), there is always the 64-bit generation coming up! And coming up quicker than you might think, by the way.

(In fact, 1987 will be the *first* year in which 16-bit micro sales will exceed 8-bit micro sales—and then by a narrow margin! Watch the second half of the year. Unit sales volume of 68020 machines and 386 machines will not match those of 68000 and 286 machines until 1991 at the very *earliest*. And yet all one reads about is the 68020 (or 030) and the 386. Well, rock shooting toys aren't fun to write about even if they *do* move lots of unit volume.)

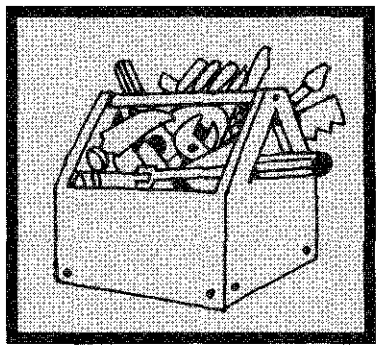
I prefer writing about the higher-performance devices even though my livelihood in the next couple years is most certainly going to be based on sales to rock-shooting toy owners. The food editor of your local metropolitan newspaper likes to write about fancy French restaurants, knowing full well that the volume and profits are at the Golden Arches.

David Bunnell, in his latest *PC World* column, wonders why the terms of the U.S.-Japan semi trade agreement were kept secret. I wonder who ties David's shoelaces in the morning? □


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

**TSF** is owned and operated by programmers, so we understand your needs. We believe *and practice* integrity in all business dealings. We advertise real prices and offer our best price to all customers. We provide prompt delivery of current product versions.

We accept checks, Visa, MasterCard, American Express and COD. We charge your card only when we ship and do **not** add a surcharge. We provide free UPS delivery on orders over $100 ($3 delivery charge for small orders). Our COD fee is $4. We allow return privileges on most products. ***Give us a try!***




**The Software Family**
649 Mission Street
San Francisco, CA 94105


See us at the Computer Faire, Moscone Center, San Francisco, March 26 through 29, 1987!

## Winter Specials

### Basic Language

| | |
|---|---|
| Microsoft Quick Basic (List $99) | $65 |
| MicroHelp Inside Track ($65) | $55 |
| MicroHelp Mach 2 (List $75) | $60 |
| MicroHelp Stay-Res (List $95) | $75 |
| Sterling Castle Tools (List $99) | $80 |

### C Language

| | |
|---|---|
| Datalight C (List $99) | $75 |
| Microsoft C w/Codeview ($450) | $270 |
| Guidelines C++ (List $195) | $175 |
| Lattice C (List $500) | $270 |
| Mark Williams C (List $500) | $290 |
| Rational Instant C | $375 |
| Gimple C-Terp (List $300) | $224 |
| Run/C Professional (List $250) | $160 |
| Run/C Interpreter (List $120) | $85 |
| Gimpel PC Lint (List $139) | $103 |
| Softcraft Btrieve (List $245) | $190 |
| Lattice dBase III Intfc ($250) | $175 |
| Blaise View Manager (List $275) | $197 |
| Vermont Window for Data ($295) | $259 |
| Lattice Curses (List $125) | $90 |
| Essentials C Essentials ($100) | $80 |
| Essentails Graphics (List $250) | $200 |
| Blaise C Tools+ (List $175) | $125 |
| Essentials Utility (List $185) | $130 |
| Greenleaf Functions (List $185) | $130 |
| Greenleaf Comm (List $185) | $130 |
| Blaise Async Manager ($175) | $125 |

***... and many more***

### Pascal Language

| | |
|---|---|
| Microsoft Pascal (List $300) | $180 |
| Blaise View Manger (List $275) | $197 |
| Blaise Async Manager ($175) | $125 |
| Blaise Pascal Tools+ (List $175) | $125 |

### Turbo Pascal

| | |
|---|---|
| Turbo Pascal BCD & 8087 ($100) | $65 |
| Software Channels Alice ($95) | $70 |
| Kydor Symbolic Debugger ($49) | $40 |
| Blaise Turbo Async ($100) | $80 |
| Blaise Power Tools ($100) | $80 |
| Borland Turbo Editor (List $70) | $48 |

### Other Tools

| | |
|---|---|
| Custom Software PC/VI ($149) | $119 |
| MKS Toolkit ($List $139) | $115 |
| Periscope I (List $295) | $235 |
| Periscope II (List $145) | $108 |
| Phoenix Plink+ (List $495) | $315 |
| Phoenix Pfinish (List $395) | $230 |
| Phoenix Pfix86+ (List $395) | $225 |
| Polytron Make (List $99) | $80 |
| Quilt SRMS + Qmake (List $199) | $165 |
| Seidl SVM + SMK (List $379) | $330 |
| Dan Bricklin's Demo (List $75) | $68 |
| Opt-Tech Sort (List $149) | $126 |
| Borland Turbo Prolog (List $99) | $65 |
| Microsoft MASM (List $150) | $98 |
| Microsoft Fortran (List $350) | $200 |

***... call for price list***

**Basic:** ***Mach2*** from ***MicroHelp*** expands Basic's horizons with window management, input editing, array sorting and print formatting. It also provides MSDOS/BIOS function calls and allows you to break the 64K data barrier. Written in assembler for high performance. For Bascom, Quick Basic , BASICA and GWBASIC. (List $75) Only $60 from TSF. 30 day money back guarantee!

**C:** ***Gimpel's PC-Lint*** evaluates your source code to locate insidious C bugs like "=" instead of "==" and mismatched function parameters. Evaluates multiple source files to validate use of all public functions and variables. Compatible with all popular compilers. 30 day trial period (15% re-stock fee). (List $139) Only $103 from **TSF.**

**All Languages:** ***Periscope II*** speeds debugging with multiple windows, symbolic memory addressing, and sophisticated trace and breakpoints. Includes an NMI break switch so you can interrupt programs at any time without pre-setting break points -- ends guesswork for hard-to-locate crashes. (List $145) Only $108 from **TSF.** 30 day money back guarantee!

**Turbo:** ***Kydor's Symbolic Debugger*** provides symbolic debugging for Turbo Pascal programs. Includes trace, breakpoint, memory display and memory modification. Runs from within Turbo's environment, so you don't loose any Turbo features. Written in assembler for speedy operation and low memory overhead. (List $49) Only $40 from **TSF.**

**Communications:** ***RamNet*** from ***System Design*** provides background file transfers and e-mail for your PC. Incoming and outgoing calls are processed automatically while you continue with your normal work. Outgoing calls can be scheduled to take advantage of late night phone rates and to concentrate messages in a multi-node ***RamNet*** network. (List $149) Only $129 from **TSF.** A two node starter kit is only $248. 30 day money back guarantee!

QUILT MICROSOFT BORLAND INTERNATIONAL Lattice Plumage BLAISE COMPUTING



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Starting Your Own Software House

by
**Ray Duncan**

Are you a talented, productive programmer who dreams of being his own boss? When you page through *BYTE* and see advertisements from multi-million-dollar companies that started with one or two people's midnight coding, do you feel a tug at the ol' heartstrings? Are you tired of working set hours or grinding out pieces of code to someone else's specifications? Do you have an idea for a new piece of software that you think might set the world on fire and that you'd like to publish yourself? Read on.

As a programmer working for a typical medium to large American business, you have probably felt a little smug when you considered the administrative people who worry about the day to day details of keeping the company alive (if you gave any thought to those people at all): After all, "they aren't creative." They are the drudges, you are the artist, right? (Most companies in fact humor this viewpoint and are far more lenient about working hours, dress, and "attitude" with programmers than they are with any other employees. Although programmers rarely appreciate the fact, their positions are relatively unstructured and carefree compared to, say, a sales manager's.)

Are you prepared to move from a steady job, where you get paid to do what you enjoy—programming—to an environment where you will initially spend a very large fraction of your time:

- writing flyers, press releases, catalogs, and manuals
- searching for the elusive Holy Grail: a reliable, prompt print shop with reasonable prices
- selecting magazines for ad placement
- getting ads designed, typeset, and to the magazines prior to the deadlines
- checking to make sure that the ads appeared
- stuffing envelopes until 3:00 am in response to magazine "bingo cards"
- trying to contact magazine editors and other bigwigs (who don't return your calls)
- selecting, ordering, and keeping inventory on binders, diskettes, disk labels, manual index tabs, page lifters, and the other paraphernalia involved in the manufacture of a professional-looking software package
- keeping track of a score of different taxes and license fees and paying them when due
- paying the bills when they are due (and not any earlier!)
- worrying about meeting the next payroll
- fending off unscrupulous salesmen of every description and phone extortions from "Public Servants" for donations for the Policeman's Ball and similar events.

Are you a highly motivated, "self-starting" person who can set and work towards long-term goals without getting diverted? Are you prepared to give up your evenings, weekends, and vacations for the three to five years it can take a small business to become stable and profitable? If you have a wife and/or family, are they going to be willing to go without little luxuries, and especially without seeing much of you, for an indefinite period of time? Are you ready for the "reality shock" of being dragged away from your autistic little world of total control over a computer, where bits are always either on or off, to the larger world of business relationships where nothing is completely under your control and issues are rarely black and white? Will you be able to cope if your fledgling business eventually fails? (The majority of small businesses in this country *do* fail in the first five years of their existence.)

Still interested? Then let us begin. Running your own software house can and will include all the stresses and aggravations that I have listed above (and more besides). But it is also interesting, entertaining, a chance to meet talented people from many different walks of life, the freedom to learn something new every day, and a tremendous opportunity for (what the psychologists are fond of calling) personal growth.

This article is the first of a series in which I hope to present, in a readily digestible form, some general guidelines for the aspiring software entrepreneur. The topics we'll discuss in the coming months include

- Taxes & Licenses
- Supplies, Inventory, and Printing
- Advertising
- Reviews, Press Releases, and other Free Ink
- Handling Inquiries
- Documentation and Packaging
- Technical Support
- Newsletters and Updates
- Employees, Payroll, and Insurance

I'm open to discussion and suggestions for other topics, too. Since *Programmer's Journal* has a relatively short lead time for articles, it's possible and desirable for your letters to affect the direction this series will take.

## Credentials

Lest you think me too presumptuous, let me tell you what experience I am basing this series of articles on. I learned most of what I will write about the hard way: When I started my own software company, I didn't know any more about running a business than I did about building a space shuttle (I still know next to nothing about space shuttles). I've made a lot of expensive mistakes along the way too—though fortunately, none of them turned out to be lethal to the company.

I worked my way through college and medical school consulting and writing programs for clinical laboratories (reference labs that perform blood tests that are too exotic or low volume for the laboratories



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

found in doctor's offices or hospitals). The real-time computer systems in these labs were typically built around minicomputers such as the DEC PDP-11 or Raytheon 703. I had been doing this for about seven years when the first reasonable microcomputers appeared, and I soon became the proud owner (ca. 1977) of an Imsai 8080 with an ICOM floppy disk and an old Teletype. I had the idea of putting together an inexpensive specimen tracking and billing system for small clinical laboratories based on microcomputers and formed a company named Laboratory Microsystems (LMI) for that purpose. This original idea never culminated in a marketable product although I had a lot of fun with the Imsai.

About three years later, I was working on a project that involved data acquisition from patient monitors in an intensive care unit and was looking around for an interactive language that was fast enough for hardware control. We were using Cromemco Z-80 microcomputers under CP/M and MP/M at the time, with one of the earliest 5.25" Winchester hard disks. After surveying the available languages, I decided to roll my own and brought up a Z-80 FORTH system under CP/M 2.2 (using a public domain version of FORTH called FIG-FORTH as a starting point) to do the project. At that time, Forth systems traditionally ran standalone or "native" and didn't use file systems. My version ran under CP/M as a task and used normal CP/M files for storage, and I thought that this was innovative enough that someone else might find it useful too.

**I learned most of what I will write about the hard way.**

I started out by placing some very small classified advertisements in *Computer Shopper* in the summer of 1980. These advertisements were practically free and resulted in a few sales and enough letters of inquiry and telephone calls to lead me on. I then bought a few small display ads in *Dr. Dobb's Journal*, and when a chunk of cash came in from a consulting job, I squandered $1000 of it for a quarter-page advertisement in *BYTE*, January 1981. I sold Z-80 FORTHs by mail and telephone in my spare time for about a year and a half, using an answering machine and answering service, and packaging the software, typing invoices, and stuffing envelopes into the wee hours of the morning. By that time I was getting pretty weary of being just another rat in the Skinner Box of academic medicine, so I left my position at Cedars-Sinai Medical Center, rented an office, and donned a new persona as a full-time software vendor.

Today, LMI sells Forth interpreters, compilers, and cross-compilers for a broad variety of microprocessors. We also do consulting work on occasion in Assembler, Forth, and C. During the last few years, the company has ranged in size from two to five full-time employees, along with several additional part-time or freelance programmers. The Forth market will never be huge, but it's a living.

### What's Your Product?

I suspect you wouldn't be reading this article if you didn't already have a fairly detailed mental blueprint for the program you will be selling. Your potential product probably falls into one of three categories:

Category 1: Stroke Of Genius Program. Fills a need that no one perceived before and creates a whole new market segment. Doesn't happen often (though Dan Bricklin has done it twice now), but has huge potential for riches, fame, and glory.

Category 2: Better Mousetrap Program. Your program will enter an already crowded market segment where the price per unit is in the $50 to $500 range, but is significantly faster, easier to use, more flexible, or has novel features not offered by the competition.

Category 3: Price Is No Object Program. Your program targets a highly specialized, vertical market and will sell for a very high unit price, perhaps $5000–$10000 per copy or more. You can write this program because, unlike most people, you can combine overwhelming expertise in the application area with programming proficiency.

If your program falls into categories #1 or #3 and you pull off a good implementation, you don't need any advice from me—the world will beat a path to your door anyway. It's the fellow in category #2 that I intend to address. In any event, this series isn't about designing and writing your program, it's about all the other considerations of surviving while trying to sell the program once it is written. I would like to remind you of two time-tested rules of thumb, though:

1. Write about something you know.

2. Know your competition.

You will be running this business on your own time, for a change, and time translates directly to money. If you have to spend



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# PORTABLE POWER WITH MasterFORTH

Whether you program on the **Macintosh**, the **IBM PC**, an **Apple II** series, a **CP/M** system, or the **Commodore 64**, your program will run unchanged on all the rest.

If you write for yourself, MasterFORTH will protect your investment. If you write for others, it will expand your marketplace.

Forth is interactive – you have immediate feedback as you program, every step of the way. Forth is fast, too, and you can use its built-in assembler to make it even faster. MasterFORTH's relocatable utilities and headerless code let you pack a lot more program into your memory. The resident debugger lets you decompile, breakpoint and trace your way through most programming problems. A string package, file interface and full screen editor are all standard features. And the optional target compiler lets you optimize your application for virtually any programming environment.

The package exactly matches *Mastering Forth* (Brady, 1984) and meets all provisions of the Forth-83 Standard.

| | |
|---|---|
| MasterFORTH standard package | $125 |
| (Commodore 64 with graphics) | $100 |
| Extensions | |
| Floating Point | $60 |
| Graphics (selected systems) | $60 |
| Module relocator (with utility sources) | $60 |
| TAGS (Target Applic. Generation System) – MasterFORTH, target compiler and relocator | $495 |
| Publications & Application Models | |
| Printed source listings (each) | $35 |
| Forth-83 International Standard | $15 |
| Model Library, Volumes 1-3 (each) | $40 |

**(213) 821-4340**

MICROMOTION

8726 S. Sepulveda Bl., #A171
Los Angeles, CA 90045

Number 13 on the Reader Service Card


months doing research into a subject before you can even write a credible program, you are going to starve. If you don't know the competition, you will waste valuable time making mistakes that other people have already made for you or produce a program that is missing features that your potential customer considers standard or indispensable.

### Getting Started

Unless you've got massive savings laid away, or you've got an independent source of income, it may be wiser to phase into your new business over some period of time. You can work nights and weekends on your own software while continuing to put in an honest day's work at your current employer's and not have to worry about the rent. When your program is nearly ready to sell, allow about a month to complete the following arrangements, visiting the various offices and filling out the forms in your spare time.

**Your Company Name:** Choose a name for your company that isn't cute or pretentious and that doesn't include your own name. This may help things go more smoothly during the initial period when no one has heard of you before. Having your own name in the company name is a sure tip off that you are a small-time operation; while we wouldn't want to mislead anybody, there are times (you're trying to open an account with a supplier) when it pays to be circumspect.

Then go to your local city hall or county clerk's office and file the equivalent of Los Angeles's Fictitious Business Name Statement. Banks require proof that this has been done before they will let you open an account under a name that is not your own. There is usually a small fee ($10 in Los Angeles), and you may also be required to publish a notice in a public periodical. It turns out that in California there are small daily journals that exist just for this purpose, and you will find information on how to reach them wherever you file the Fictitious Name Statement.

**Bank Matters:** The fictitious name application completed, go to your favorite bank and open a checking account in the business name. If you have had your personal account at the same branch for a long time, go there. If not, ask around and find a bank in your area known for its willingness to work closely with small businesses. Make an effort to get acquainted with an assistant manager or other bank officer at the outset. Don't wait until you have a problem with an account or need a loan to get chummy with the people who can solve your problems. It also helps to get on good terms with the merchant teller, who is usually senior to the other tellers; if you should become temporarily overdrawn, he or she may be willing to exercise the bank's option to cover you, or at least call you and tell you to bring in some money to clear the check. Understandably, if they don't know your face, they'll just bounce your checks without further ado.

Once you get your checking account, you will also want to enter into a Visa/Mastercard Merchant Agreement with your bank. This is not to get a Visa/Mastercard of your own, but to make it possible for you to accept credit card orders from your customers. Credit card slips are deposited into your checking account just as checks are, but they are better than checks! Credit card sales of some arbitrary size (usually $50 or $75) must be called in for authorization, but you can call them *all* in for authorization if you wish. Once you have been issued an approval number and have written it on the credit card slip, that slip is as good as gold: Unlike a check, it can't bounce!

Here is another occasion where you may have to be a bit judicious when describing your business to outsiders. Unless your product is a hit and is picked up by Softsel or other major distributors, you will probably be doing most of your business by phone or mail. Banks are justifiably leery of mail order operations, due to their grim experiences with mail order fraud (for example, an Arizona company called World Power Systems took thousands of computer hobbyists to the cleaners a few years back). If you tell your bank, or they suspect, that you are going to be primarily a mail and phone order business, they will almost certainly *not* be willing to let you open a credit card merchant account—unless you have known them for a very long time and they have reason to feel confidence in your ethics and longevity.

If you can't get a credit card merchant account, you will have to make arrangements with a friendly computer store or other small business to put through your credit card charges because a complete inability to accept credit card orders will significantly cut into your phone sales. Needless to say, this alternative is highly undesirable, and you should continue to campaign with bank officers at every opportunity to get your own account.

**Licenses and Such:** First, you will need a city business license. In Los Angeles, this



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

is a thinly disguised local *tax* since it is based on the company's gross revenue for the previous year. The first year, the city will just charge you some arbitrary amount and nail you for the remainder the *next* year. If you fail to get the license, most cities have ways of eventually catching up with you (by inspection of state income tax filings and so forth) and are likely to levy a stiff penalty. Be careful when you describe the nature of your business to the city. The tax on revenues obtained from retail sales is usually at a lower rate than that on income derived from consulting. If the city gets the idea that you are in the service (consulting) business, the burden of proof will be on you to prove otherwise, leading to a lot of paperwork and annoying encounters with roving City investigators.

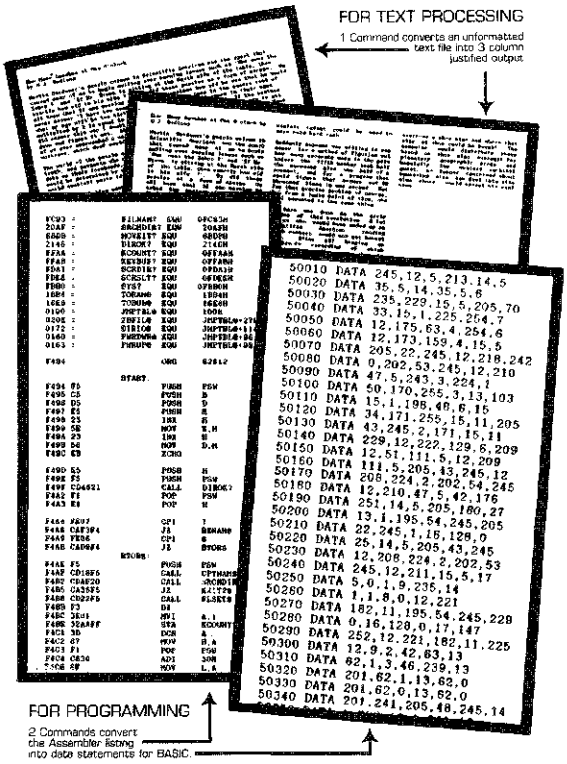
You will also need a Federal Employer ID number at some point, even if you plan to pay only yourself, so you might as well apply for it early. When large companies send you a purchase order, they will require you to supply this number; otherwise, they are obligated by the IRS to withhold a portion of their payment to you for tax purposes. You can pick up an application at your local IRS office and file it by mail; the ID number will take a few weeks to come through.

If your business is located within a state with a sales tax, you must collect that sales tax on each package you sell to a customer within the same state. In California, the taxes are passed on to the state quarterly, and the whole process is administered by an agency amazingly named the "State Bureau of Equalization." Once you have your bank account and Fictitious Name Statement and *before* you make a single sales transaction under your company name, head for the local office of your State's equivalent of the State Bureau of Equalization and apply for a Resale Number. They will likely require you to make a deposit against your first quarter's taxes. This deposit will be scaled to your projected income for the quarter, so quote them a low number! In California, you have the option of buying a bond at your bank which is made out to the state and serves as the deposit, but the interest on the bond is credited to your own checking account.

Be sure to file your quarterly sales tax forms with payment in a conscientious and prompt manner. If you take a careful look at the form before you sell anything and note the information you will be needing, it will save you a lot of time later. Sales out of state or overseas are not taxed, and sales to dealers or distributors for further resale are not taxed. On the other hand, in California several of the counties have additional 0.5% or 1% sales taxes that are used to finance rapid transit projects. Consequently, careful records of the destination of each package shipped are incredibly important so that you pay just the right amount of sales tax—and no extra.

These careful records will be even more vital if your sales taxes are ever audited. The California State Board of Equalization is a powerful and fearsome arm of the Law, with a somewhat fascist mentality—worse by far in my experience than the IRS. They can tie up nearly unlimited amounts of your time with paperwork and demands for old records, or they can shut you down at a whim by revoking your Resale Number. If they catch you in even an innocent error, they may harass you for years afterwards. Try to do everything on time and shipshape so that you never come to their attention.

**Business Input/Output:** It's probably wise to open a post office box as quickly as possible and make sure that address appears in all of your ads and correspondence from the outset. This will eliminate a lot of mail forwarding hassles if your business expands rapidly and requires changes of loca-


**CUT HOURS OFF TEXT PROCESSING TIME WITH:**

# TEXT-PRO

Use Text-Pro to make reusable, canned multiple Search & Replaces to use on different files.

Create Batch files, reformat, select, merge, split, sort files, insert records, do Global Search & Replace without going into the editor.

Convert variable to fixed length records & vice versa.
Produce documentation with real impact.
Format listings.

Makes light work of Code & Data Conversions—handles Code conversion problems with a single command.
Ideal for porting Code.

Chain several of Text-Pro's 29 programs together to produce dramatic changes in Text files.

Use to speed programming, conversion, text editing.

In use in major Electronic Publishing firms & Software Houses.

For MSDOS & PCDOS machines.

**King Computer Services, Inc., 1016 N. New Hampshire**
**Los Angeles, CA 90029 (213) 661-2063**

Number 91 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tion. If you run your business out of your house, it will also keep customers from showing up at your front door when you least expect them.

You will also need to make connections with at least two shippers, probably UPS and an additional overnight carrier such as Federal Express. You don't want to ship any of your precious brainchildren *out* by the U.S. Post Office unless there is absolutely no alternative.

Getting started with UPS is somewhat formal. You will need to call their local office and make an appointment with one of their customer representatives. This fellow will come out to your house or office and ask some questions about the nature of your product (for insurance purposes) and how many packages you expect to be shipping per week. You will have to sign an agreement and pay a deposit (probably in the range $100–$500) based on your estimated shipping volumes. He will then give you a starter supply of each of the UPS forms and issue you a little rubber stamp with your account number on it. The UPS driver will come by daily to pick up or deliver, and UPS bills you on a weekly basis for its services. You can save up the bills and pay them once a month, but if the monthly total consistently exceeds the amount of your initial deposit, you will be asked to fork over some more deposit money.

Opening an account with Federal Express is easy; simply call up one of their service representatives, give them the necessary information, and a set of supplies (preprinted airbills, overnight envelopes, etc.) will be shipped out to you within a week or so. Federal Express is nice to have on hand as an alternative to UPS because they will deliver or pick up on Saturday, you can charge shipments to the recipient's or a third party's account number when necessary, and they are much more flexible about pickups. One disadvantage compared to UPS: Federal Express does not accept C.O.D. packages.

**Phones:** Get another line installed right away, putting the number in the name of your new company. This will help people reach you through Information, allow you to transfer the business number to new quarters as your business grows, and permit you to *ignore* the business phone after 5 pm. By the time your first advertisement appears, you should either be in a position to answer the phone during normal business hours or have someone else that can do it. You may be able to find a small secretarial service that will take orders and messages for you for a monthly fee of $250 or so; this is an economical temporizing measure until you need full time help and will give a much better impression to potential customers than an answering service (which should be your next to last resort). An answering machine should be the absolute last resort—but more on the topic of phones in a later installment.

**Outside Advisors:** One of the smartest moves you can make is to seek expert help in areas you know nothing about. I recommend that you engage the services of a lawyer and a CPA at an early stage. Ideally you would already be acquainted with both these people either personally or professionally and have complete faith in their integrity and competence (one exception: *Don't* select relatives!). That failing, ask around for recommendations from friends, colleagues, or other small businessmen. Schedule a brief interview with both the legal eagle and the accountant, during which they can get to know you and ask pertinent questions about your new business.

As need arises, the lawyer will be available to help you review contracts, nondisclosure agreements, and other legal entanglements, file trademark registrations and articles of incorporation, and advise you on relationships with employees, suppliers, and clients. The CPA can guide you on the tax structure of your business (sole proprietorship, partnership, corporation, etc.) and notify you when it is appropriate to change this structure; he will fill out state and federal income tax forms, run interference in case of an audit, draw up financial statements if you need to apply for a loan, and give you sound advice on payroll, lease vs. purchase of equipment, renting of office space, and a zillion other subjects. The total fees for these services from a lawyer and a CPA for a typical small business will not exceed $2000-$3000 per year and will be money well invested. □

**References**

Here is an assortment of books that contain useful tips for the would-be software mogul:


*Growing Concerns: Building and Managing the Smaller Business*, a collection of 36 articles from the *Harvard Business Review*, edited by David E. Gumpert. John Wiley & Sons, New York, 1984. ISBN 0-471-88677-7.

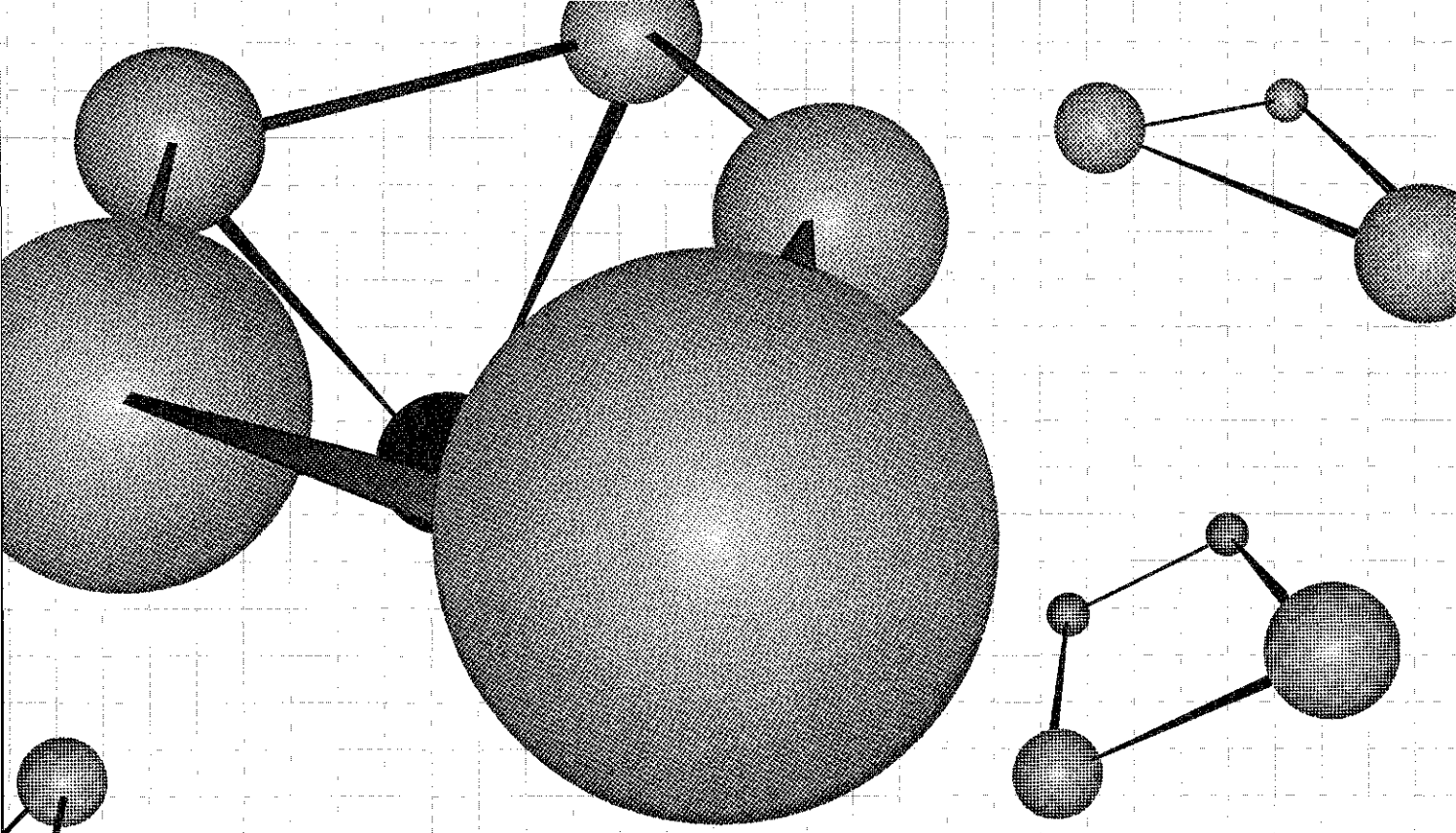
*What They Don't Teach You at Harvard Business School* by Mark H. McCormack. Bantam Books, New York, 1984. ISBN 0-553-05061-3.

*How to Become Financially Successful by Owning Your Own Business* by Albert J. Lowry. Simon and Schuster, New York, 1981. ISBN 0-671-41261-2.

*Legal Care for your Software* by Daniel Remer. Nolo Press, Berkeley, 1982. ISBN 0-917316-85-1.

*Getting to Yes: Negotiating Agreement without Giving In* by Roger Fisher and William Ury. Penguin Books, New York, 1983. ISBN 0-14-00-6534-2.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



# Aztec C - the *best* C in the Micro-cosm

## Why is Manx Aztec C the best C ?

Because Manx offers a combination of performance, features, and pricing that are unbeatable.

What kind of performance ? In the 2/87 issue of Computer Languages, our 3.2 release of Aztec C for PC-DOS turned out 109,489 iterations of the Dhrystone benchmark per minute to108,303 for Microsoft C. Our 4.1 code is even faster.

What kind of features ? Features like in-line assembly code, a professional make utility, library source, ROM support, debuggers, assemblers, and overlay linkers. Many of these features are unavailable or extremely high priced options in other systems. Microsoft, for instance, does not provide library source or in-line assembly.

What kind of pricing ? For $199, the Professional version of Aztec C gives you an Optimized C, a macro assembler, a librarian, overlay linker, extensive UNIX and MS-DOS specific library routines, a powerful debugger, sample programs, and over 500 pages of documentation. Manx Aztec C has a hard earned and well deserved reputation as a leading C software development system for PC-DOS and MS-DOS. At this price, it is also one of the best software values available.

Over 80 percent of our sales are repeat sales or are based on recommendations. That says it all. At Manx, we extend ourselves to the limit to provide the best software, the best service, and the best support possible. Call today, and join the over 50,000 users who know the Manx difference.

Manx also provides C systems for Macintosh, Amiga, CP/M-80, TRS-80, Apple //, 6502/65C02 ROM, 8080/Z80 ROM, 680x0 ROM, and 8086/80x86 ROM. **Call for prices and information.**

## Aztec C86 4.1

**PC-DOS • MS-DOS • ROM**

**Aztec C86-p Professional System .....$199**

• optimized C with near, far, huge, small, and large memory • in-line assembler • in-line 8087/80287 • ANSI support • Fast Float (32 bit) • optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • MS-DOS 3.x file sharing & locking • comprehensive libraries of UNIX, MS-DOS, Screen, Graphics, and special run time routines.

**Aztec C86-d Developer System.....$299**

• includes all of Aztec C86-p • Unix utilities make, diff,grep • vi editor • 6+ memory models • Profiler.

**Aztec C86-c Commercial System.....$499**

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 cross dev. option • One year of updates.

## How To Become An Aztec C User

To become an Aztec C user call **800-221-0440.** From NJ or international locations call **201-542-2121. Telex: 4995812** or **FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available.**

Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Systems are upgradable by paying the difference in price plus $10. One and two day delivery is available for all domestic and most international destinations.

**To order or for more information call today.**

**1-800-221-0440**

**In NJ or international call (201) 542-2121 • TELEX: 4995812**

MANX

Manx Software Systems
1 Industrial Way, Eatontown, NJ 07724

REPRODUCED 2004 BY UNZ.ORG
ALL RIGHTS REPRODUCTION PROHIBITED

Number 92 on the Reader Service Card

# Pink Noise

by
**Frank D. Greco**

**It's a Breath Mint . . . It's a Floor Wax . . . No, It's Both!**
Look for IBM to announce the availability of their PL.8 compiler for the RT PC. In true IBM style, being all things to all people, the PL.8 compiler accepts multiple source languages and produces multiple object codes. You heard that right! As originally described in the June 1982, Proceedings of the SIGPLAN Symposium on Compiler Construction, the PL.8 compiler accepts Pascal, PL.8 (a PL/I variant for systems programming), and C language source code and produces object code for the System/370, Motorola 680X0, 801 (precursor to the ROMP chip in the RT), two other experimental IBM processors, and now Intel 80X86 processor families. The basis of the PL.8 compiler is in the translation of the source language into IBM's assembly-language-like IL (intermediate language), which is inspected for optimizations and subsequently converted to the target machine language.

**No Man is an Island**
But a man and a woman started one. Steven Dompier's and Diane Ascher's Island Graphics of Sausalito, California, was just infused with needed capital from international giant Philips. Philips Medical Systems acquired ten percent equity and secures a seat on the Island board of directors. Note that Philips was one of the first customers of Pixar, the Lucasfilms spin-off in which Steve Jobs has some equity. Under the Philips agreement, Island's sophisticated painting software, which was popularized on Sun Microsystems' color Unix boxes, will run on the superpowered Pixar graphics machine. If you have never seen the Island graphics color 3-D paint package, here's a rough analogy: Island graphics paint software is to MacPaint as a memory-mapped screen editor is to EDLIN.

The Island natives are also collaborating with MicroPro International to give the WordStar makers their first desktop publishing product. From the rumors I've heard, it's a monster!

**Dana Da Destroya**
Former Convergent Technologies founder Allen Michels has started a new company called Dana Computer. Along with about three dozen other companies, Dana Computer hopes to capitalize on the upcoming lucrative workstation market. However, their desktop dynamo is expected to perform 6 megaflops and contain multiple processors. This new machine, called the Titan, is based on the new 16 MHz RISC chip from MIPS Computer, Inc., and is expected to turn in a scalar performance of 10 to 30 MIPS. In addition, its radical architecture will have a superfast (50 megapixel per second) graphics display integrated with the CPU. Remember, this is all on a single-user desktop machine—in other words, a PC! However, initial pricing will see the Titan around the non-PC price of $75,000 to $100,000.

**Wyse Move**
Wyse Technology, Inc., (1-800-GET-WYSE) the premier OEM supplier of computer terminals, reduced the price of all their ASCII terminals to compete with the new IBM ASCII terminals (which are so braindamaged that . . . but that's another story). If you want a "vanilla" computer terminal, these high quality terminals are perfect for plugging into any Unix box.

**Another Wyse Move**
The price of the Wyse 700 high resolution (1280 x 800 pixel) interlaced monitor/board combination was dropped drastically to $999 list! I've seen retail prices around $700 already. Lotus drivers (the software kind!) for the Wyse 700 are available for only $25 from Phoenix Technologies, Ltd., of Norwood, Mass. When a vendor does something like that, a new product announcement is in the offing. My guess is a newer non-interlaced high resolution monitor at the same price point as the original list of the Wyse 700.

I'm also very familiar with the Wyse PC product line and would overwhelmingly recommend them to anyone who needs an extremely compatible PC or (1 wait state 10 MHz) PC AT clone priced very competitively and looking great! The Wyse people have always paid attention to styling, which is now paying off. Just look at how many people are cloning the Wyse models 50 and 60 computer terminals. Cloning is the sincerest form of flattery (or something like that).

As far as the company goes, President Phillip White, who formerly spun his magic for the fast-growing Altos Computers, has been quoted as saying that Wyse will expect a greater percentage of revenues from the PC market in the financial quarters to come. As more and more users are turning to PCs to act as intelligent terminals, Wyse is moving in that direction. And since Wyse also owns monitor-maker Amdek, I would expect the Wyse Guys to make some *big* moves in the PC arena. From pals on the west coast, I understand their 386 machine smokes!

**I Want My V/AT**
Wyse competitors Televideo Systems, Inc., recently acquired an equity stake in Microport Systems, Inc. Microport is the


PRODUCED 2007 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

company I've been telling you about that sells a super low cost and highly disk-efficient version of System V Unix for the IBM PC ATs and compatibles. Microport, if you remember your history, was a spin-off from Digital Research, Inc., when DRI decided that it no longer wanted to be in the Unix System V porting business. Microport sells a minimal runtime version of AT&T System V Unix, called System V/AT for $159. Apparently, Televideo is expecting to be in the Unix business soon. Since Microport has told me that they have a 80386 port coupled with Locus Computing's Simultask product, I would bet that Televideo has an imminent 80386 announcement (boy, is the 386 getting to be a "safe" prognostication nowadays or what!).

**Screen Gem**

Radius (408-732-1010) of Sunnyvale, California, has a fantastic new product for the upcoming desktop CAP (Computer Aided Publication) market. It was designed by digital electronics whiz and designer of the Apple Macintosh's motherboard Burrell Smith. Radius is offering a large screen, relatively high resolution (640 by 864) vertical tube monitor for the smallish Macintosh. In addition, the Full Page Display (FPD) maintains (basically) the same visual registration as the regular Mac screen and the sexy LaserWriter. What most potential publishing clients do not like about the Macintosh is its miniscule screen. What most potential publishing clients do like about the Macintosh is its wealth of super-functional and highly-usable application programs (especially items like Aldus's Pagemaker). As long as you have the new Macintosh ROM and $1995, you can now finally see a *full* page of Pagemaker in action. Halleluja! So for about $10,000, you can set up a very usable small scale publishing shop. Or at least, your own in-house publishing group.

Rumors abound in Austin, Texas, that Apple Computer has been talking to Texas Instruments about using their NuBus as the open-bus architecture for the upcoming Apple Unix workstations projected to be announced in 1987. The NuBus is a full 32-bit synchronous bus currently used only by AI gurus Lisp Machine in Cambridge (and of course, TI itself).

**Megabits. . . Breakfast of Champions**

Someone finally announced a monitor that matches the resolution of the low end laser printers. MegaScan Technology, Inc., (412-443-5820) offers the UHR-3000 monitor that can display 300 dots per inch on-screen. So OEMs can now build a true visual representation of what was sent to the laser printer for previewing purposes. To handle the tremendous overhead of spitting the multiplicity of bits to the monitor from 2- to 3-megabyte video memory, MegaScan also has a frame buffer display processor called the PDP-3000 (based on the Motorola 32-bit 68020 and the OctRop raster processor) that handles the high resolution at incredible speeds. We're talking raster operations at 128 megapixels per second folks! Just add a PostScript driver and bingo, you have a high resolution laser printer previewing product for the printing industry (most "mom-and-pop" shops will settle for a Mac with a LaserWriter). But unfortunately, these high-end products are priced in the high-end stratosphere. The UHR-3000 goes for $5,500, and the PDP-3000 sells for $11,000.

**CD-rama**

Hitachi has announced a relatively inexpensive CD-ROM drive for the PC market. The CDR 2500 552-Megabyte CD-ROM drive lists for $940 and includes the interface board, a cable, and a demo disk. I imagine they will be showing this product at the next Comdex.

**Warning: Atarians Seen Approaching the Earth!**

Atari announced strong intentions of going public very soon by issuing a prospectus on the company. By the time you read this, Jack Tramiel and company will probably be listed on the OTC market. In the prospectus there is mention of a future multi-user, multi-tasking 32-bit machine. Could this be their Unix machine? Friends at AT&T are telling me they can't say or they will lose their jobs, but talks are definitely continuing with Jack's Pack. Also, strong rumors have machines in the Atari labs running Alpha versions of the rumored "Future" or "Advanced" multi-tasking MS DOS (a.k.a MS DOS 5.0).

As reported in *EET*, Atari announced the 2080 ST and the 4160 ST with, respectively, 2 megabytes and 4 megabytes of RAM. The Atarians also showed their customized graphics chip (mentioned in *PJ*, "Redirections," Volume 4, Number 5), which brings graphical performance up to par with the Commodore Amiga.

Kuma Computers, Ltd., announced a very interesting product for the economical Atari ST series. It seems that they have developed a parallel-processing board that contains 2 T414 32-bit RISC-flavored Inmos Transputers. Inmos rates the T414 at 7.5 MIPS (of course these are RISC MIPS!) and claims linear performance increases when groups of transputers are combined in a parallel network. Even if the transputer were conservatively rated at 2 (CISC) MIPS, the $2,175 Kuma board would make the ST hum along at 4 MIPS! In essence, for about



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

$3,500 (Atari ST plus the Kuma board) you have the computational equivalent of a DEC VAX 8600!

### Ichi Ban Prolog-San

A while ago in "Redirections" I reported that some Japanese grammar school and high school children are using laptop computers running Prolog to learn about computing. Just as a point of reference, the United States K-12 (kindergarten to 12th grade) school market is, on the whole, understaffed and undereducated in computer studies; a procedural language, that is, BASIC, is used to teach computer concepts. In Tokyo (as reported to me by a very close source), Sega Enterprises is currently selling the Sega AI Computer, which contains an unprogrammable runtime only Prolog interpreter (instead of a BASIC interpreter) in a 128-Kbyte ROM. Prolog is the engine for some CAI applications on cartridge. It is powered by the NEC V-20 and has access to 128K of user-RAM. Trackball, Centronics interface, 64Kbytes of video RAM, and a touch-sensitive pad are all standard. For programmability, Sega offers the Logo language.

### A Point On Counterpoint

Recently everyone from *Electronic Magazine* to *EE Times* has predicted that startup Counterpoint Computers will take over the highly prized parallel processing championship. Counterpoint claims that their Advanced Systems Platform (ASP) machines are designed so that multiple CPUs can be connected together to obtain linear increases in performance. However, I personally think they will never make winner's circle; I really think that this company will eventually crumble. Their strategy is off-base and ill-timed, and their alliances with other companies are not strong enough.

Counterpoint was started by former Convergent Technologies executive Pauline Alker back in the second quarter of 1984. It raised about $20 million in venture capital, which is small change compared to the resources of its strategic backers AT&T, Japan's Kyocera, and British & Commonwealth Shipping. Located in San Jose, California, Counterpoint has relied on hopes that AT&T would strongly back their 68020-based ASP, which uses Unix. Ms. Alker should have realized from her Convergent experiences that AT&T's computer division is still disorganized and grossly mismanaged. The recent marketing revelation from AT&T that it will concentrate more on telecommunications than on computers (*Wall Street Journal*, 10/3/86) should have demonstrated their ineptness; that was AT&T's strategy back in 1982! Millions of dollars could have been saved if they had only implemented that strategy back then. A colleague noted that if AT&T's computer division had just put their lost millions in a regular, ordinary bank account, they would have made more money just letting it sit!

According to my sources, software for the Counterpoint ASP is way behind schedule and not even up to state-of-the-market standards. AT&T was hoping to jump into the CAD/CAP graphics workstation arena with Counterpoint's fellow AT&T compatriot, Omnicad, Inc. As opposed to James Edwards' (president of AT&T's computer systems division who has some personal stake in Omnicad) forecast, the hardware/software combination was a poor match and is currently being scrapped. If Counterpoint and Ms. Alker had done their homework, they would have known that AT&T could not even market their *own* graphics workstation effectively. The phone company has high hopes (no comment) for their high resolution $5,000 AT&T DMD 5620 terminal, formerly called the BLIT, even though no one (besides a few hackers) is interested yet. Despite the lack of interest, Ma Bell is planning a cheaper version (about $2,000), which will be announced shortly. Anyone care to wager on the success of that one?

Not only that, but Counterpoint's competitors' plans for 80386-based MS DOS/Unix workstations and the ever-growing Sun Microsystems are strong obstacles to Counterpoint's success. To counter (pun intended) these opponents, Ms. Alker's current strategy depends on hopes of using Unix's multi-user capabilities in an attempt to market the ASP as a departmental computer. Other Counterpoint executives have been quoted as saying they would like to burst the perception of the company as "an engineering workstation-only vendor." Ms. Alker in *CSN*, 9/15/86: "We are a multi-user computer company that also offers bit-mapped graphics support." This is the kiss of death. She will now have to contend with additional heavyweights such as Altos, NCR, DEC, and her old employer Convergent. This battle, plus the fight with Sun, Apple, Apollo, DEC, and (dare I mention) IBM in the workstation arena, will soon have Counterpoint in serious trouble. I predict that Counterpoint will be out of business within one year.

### You Bet Your Sweet BIP-py

At the recent SIGgraph'86 conference, Meiko, Ltd., of Bristol, England, showed a 311-transputer (Inmos) version of their Computing Surface. They claim that their $850,000 machine can reach 3,000 MIPS (that's 3 BIPS!). Again, that is a RISC MIP rating, but it's still impressive. They demonstrated a graphic "ray-tracing" algorithm for the attendees. In ray tracing light rays reflected or transmitted off an object are followed ("traced") mathematically, determined to be viewable or not, and consequently displayed. If you have ever seen those super high resolution graphics of multi-colored translucent spheres, that is ray-tracing. Ray-tracing is extremely computational intensive and inherently independent, which lends it to parallel processor implementations. The Computing Surface traced 8.5 million light ray intersections in an astonishing 47 seconds!

### Microsoft Mews

Speaking of graphics, it seems that Microsoft is more than just very interested in the graphics market. They are heavily recruiting for graphics design engineers for their Redmond facilities. They want people with a minimum of 3 years programming experience in the graphics industry with 80X86 Assembler, C Language, and MS DOS expertise. EGA experience is preferred. An ad appeared in a graphics industry periodical, not a computer magazine.

The expensive (approximately $140) Microsoft *MS DOS Encyclopedia* has been recalled due to "severe errors," according to Microsoft. I called Microsoft in Redmond and was told by a sales representative that it will be re-released in six to eight months! A friend who glanced through it told me that it was relatively useless and hardly worth the money. From what I was told, it contained a lot of user-level information (for example, what the DEL command does) and not much more systems information (what we all *really* want) than their *Programmer's Reference Manual*, which sells for $40.

### See 86 Run

According to Computer Innovations, Inc., the latest version of CI-C86's C compiler compiles and executes faster than Microsoft's C compiler Version 4. I bumped into a gentleman in San Francisco during a recent Uni-OPS convention who told me that he was helping to write the upcoming CI-C86 com-



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

piler. According to him, it uses brand new compiler technology, offers near/far pointers, claims faster-than-Microsoft execution time, and will have inherent capabilities for very sexy source level debugging. Computer Innovations refuses to comment on the debugger-ability, but they did say that "It makes sense. We have to counter Microsoft's CodeView in some manner." As in the older CI-C86 C compiler system, source code to the entire library comes on disk. By the time you read this, the new CI-C86 compiler, to be unimaginatively called "C86 Plus" (ho hum) should be available. At least they didn't call it "Turbo" something or other.

### In Search Of

In the never-ending search for high quality hardware, software, vendors, and books for inclusion in this column, I've mentioned some sure winners. Off the top of my head, I can think of products and companies like the Toshiba 1100 laptop, GEOS for the C64, Toshiba's 351 printer, AT&T's TARGA boards, Metagraphics' MetaWindow, Sun Microsystems, MDS Genius monitor, Island Graphics, Adobe's PostScript, and the Wyse PC Plus and 286 machines.

Well, you can include a new one on the hit parade. Look out for Accent Software (415-949-2711) of Los Altos, California. Accent software has some of the most dazzling animation seen on a PC. As opposed to other PC-oriented graphics systems, Accent has developed customized demonstrations that feel alive! Instead of using computer programmers to design their animation, these people have gone out and used professional graphics artists to create some real killer stuff. It's about time someone put some zing into PC graphics.

I noticed this software when I happened to walk past the GRID booth at the PC Expo show in NYC. No offense to GRID, but I was stunned more by their graphics demo than by their laptop. Nancy Vanclef, Director of Marketing at Accent Software, was kind enough to send me additional samples of things they have done for other clients. All I have to say is "Wow!" For a machine not exactly noted for exciting CGA graphics, Accent has worked miracles with the lowly PC. The sad thing is that their tools for creating these customized marvels are not yet available commercially; Accent is currently creating only customized demonstrations. They are addressing the potential of commercial availability.

### Super Laser-Guided Printers

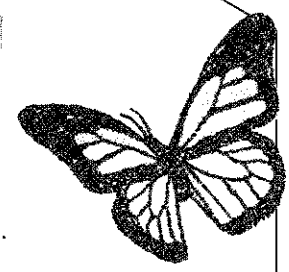
I have a late scoop on some very hot news in the laser printing business. It seems that some former Honeywell employees have devised a way to eliminate the bottleneck in a laser printer. The relative slowness of a laser printer is due to the motion of the lens/mirror mechanism that bounces the laser beam onto the light-sensitive drum. The mechanism is driven by electromechanical motor, hence the slowness. These researchers have devised a method of replacing the mirror with a hologram. By changing the quality of the hologram by electronic means (as in holographic memory systems), the light passing through the hologram (that is, the refracted light) can be tightly and quickly controlled. No moving parts alter the path of the laser! And since a hologram is a controlled transmissive device, it would be a scanner as well as a printer! To top that off, these developers say they can produce 600 dpi holographic laser engines in quantity for $100 a pop! As far as I know, these gentlemen are scouring the streets looking for some startup capital. This one could kill, I mean *kill*, the competition (Canon, Ricoh, Epson, Printronix, Mita, etc.).

Later. . . □


Chalcedony PROLOG $99.95

Not Copy Protected

A REAL Clocksin & Mellish Prolog for BOTH major microcomputing operating systems — with full cross-compatibility.



Complete with the predicates necessary for POWER AI programming:
op () name () functor () clause () =..("Univ")
...And no constraining data typing.

- Floating point
- Step-by-step tutorial
- Math functions
- Integrated editor

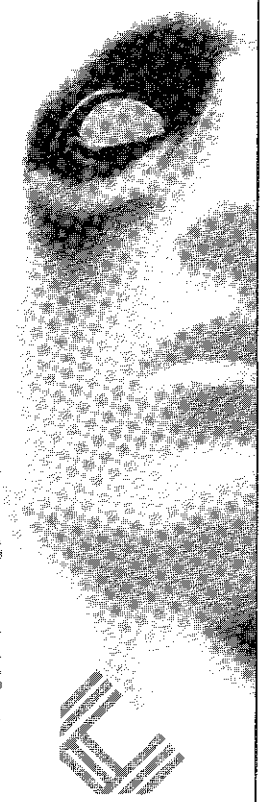
PROLOG/i

Extensible overlay library, 8087 support, large memory model (up to 640K)

PROLOG/m

Complete Macintosh environment with extensive pull-down menus and dialogue boxes.

*No Risk Offer: Examine the PROLOG/i or PROLOG/m documentation at our risk for 30 days. If not completely satisfied, return with disk still sealed for refund.*



APPLICATIONS — *Complete with SOURCE CODE*

*NFL X-pert* $49.95
A true interactive expert system written by a professional knowledge engineer. A valuable learning tool for any Prolog programmer interested in using Prolog to develop expert systems.

*PROLOG ToolBox* $29.95
More than 50 subroutines that speed and compress list handling, searches, sorts, and reversal algorithms. An inside look at the tricks of the professional Prolog programmer.

*PROLOG ToyBox* $29.95
Written by an academician to help his students understand Prolog, this collection of puzzles and mind-teasers will illustrate how the Prolog programmer creates programs that find the best solution to the problem. Turn your computer into a super reasoning machine!



CHALCEDONY SOFTWARE, INC.
5580 LA JOLLA BLVD.
SUITE 126
LA JOLLA, CA
92037
(619) 483-8513

*System Requirements:*
*Minimum 256K RAM (320K recommended) PC DOS/MS-DOS ANSI Standard Support*
*Minimum 512K Macintosh Macintosh-plus and HFS Compatible*

*SAVE 10%* when you buy either PROLOG/i or PROLOG/m and all 3 applications.

PHONE ORDERS: 1-800-621-0852 EXT 468

□ PAYMENT ENCLOSED $ _____
CA residents add 6% sales tax
□ CHARGE MY: □ MasterCard □ Visa
Card No. _____ Exp. Date _____
Signature _____
Mr./Mrs./Ms. _____
(please print full name)
Address _____
City/State/Zip _____

| | |
|---|---|
| PROLOG/i | $99.95 |
| PROLOG/m | 99.95 |
| Check: | |
| MS-DOS □ | Mac □ |
| NFL X-pert | 49.95 |
| TOOLBOX | 29.95 |
| TOYBOX | 29.95 |
| Complete Pack | 188.82 |

SHIPPING:
$ 5.00 U.S.
7.50 Canada
10.00 Carribean, Hawaii Air
20.00 Overseas Air

Number 64 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# A Graphics Print Screen Utility: a Simple Terminate and Stay Resident Program

by
**M. Steven Baker**

In a recent issue of *PJ*, I described in general the Terminate and Stay Resident (TSR) features available in MS DOS. In this article, a simple TSR program similar to the DOS GRAPHICS utility illustrates some of these features. The program PSCREEN enhances the IBM PC Print Screen Function. Similar to the DOS GRAPHICS utility, this TSR allows screen dumps in the graphics mode of the IBM Color Graphics Adapter (CGA). However, this program is configured for an Okidata 92/93 dot matrix printer (Note: not the version of the Okidata with the IBM Plug-n-Play ROMs that emulates an IBM printer). We'll also discuss briefly the graphics modes and memory map of the CGA display adapter. This programming example also illustrates some of the problems involved in converting an image from one graphics device (an IBM graphics screen) to another device, in this case a dot matrix printer.

The program listing is written in 8086 assembly language and can be assembled using MASM 4.0 under MS DOS on the IBM PC or PC clones. (I originally had equates to assemble using ASM86 under CPM86, which could be added back.) The program can be converted to a .COM file to minimize file size. Although I have written this program for the Okidata 92/93, it could easily be modified for any other graphics printer such as a Toshiba 24-pin printer or one of the laser printers with graphics capabilities. We could also change the screen graphics device to support a Hercules monochrome graphics or EGA graphics card. This simple program is easily extended to provide a graphics or text screen capture feature.

### The PC PRINT SCREEN Feature

Well, what about this PrintScreen key on the keyboard? The ROM BIOS on the IBM PC included a very convenient feature that allowed a print of the screen contents at the touch of this key. Interrupt 5 was set aside in the IBM PC as the Print Screen Interrupt. This feature functions as follows: If the keyboard interrupt routine (KB_INT in the ROM BIOS) detects that the Shift-PrintScreen key combination has been pressed, an INT 5 is generated. No keystrokes are placed in the keyboard input buffer. The ROM Bios Print Screen Function (INT 5) is invoked and determines whether the screen is in a text or graphics mode. If in one of the text modes, the entire screen contents are read from memory and written to the printer device. If the display adapter is in one of the graphics modes, a different procedure is undertaken. The screen is scanned by character cell (8 x 8 pixels in the CGA) to determine whether printable characters matching the IBM character set are present. Any matching characters are printed; otherwise, blanks are sent to the printer. In the text mode, any screen attributes are ignored.

The PrintScreen Interrupt routine uses location 50:0 (hex) as a flag location. Normally this flag byte is set to 0. While the screen is printed, this flag is set to 1. If an error occurs, this flag is set to 0ffh. If such a flag were not used and another PrintScreen key were pressed, the Keyboard interrupt would invoke the PrintScreen interrupt a second time, trashing the first printout in process.

There are some cases in which this "feature" can become a real nuisance. If a printer is not connected and Shift-PrintScreen is accidentally depressed, your application will "hang" until the printer times out and indicates an error. In some PC ROM BIOSes, the system may "hang" until you "reset" the computer. Also, IBM's choice of INT 5 for this function (and most of the IBM's BIOS interrupts for that matter) was an unfortunate decision. Intel had previously reserved interrupts 0 through 1Fh (0 to 31decimal) for internal use. The 80186, 80286 and the Nec V20/30 microprocessors issue an INT 5 if the BOUND instruction detects a range exceeded exception. The BOUND instruction was not available on the 8088/86 but can be used on these later CPUs to efficiently test whether an array subscript is within a suitable range. Compilers for the 80286 (Pascal, Modula-2, and Fortran, for example) would normally use this instruction to test array indexing. Because of this conflict, the IBM AT and most clones will execute a PrintScreen Routine if this BOUND exception occurs unless patches to the ROM BIOS are provided.

### Of Video Modes, Pixels, and Aspect Ratios

The IBM PC has available a variety of display adapters to support (either an opportunity or an inconvenience depending on your perspective). The original display adapter cards manufactured by IBM were the monochrome display using 4K of memory mapped video at segment 0B000h and the color graphics adapter using up to 16K of memory located at seg-



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ment 0B800h supporting a maximum resolution of 640 x 200 pixels. The ROM BIOS data area located from 400h to 500h (8086 segment 40h) in memory has a number of flags or locations dedicated to the video adapter installed (see the equates at the beginning of the listing).

Using information from this data area, it's possible to determine which of the two IBM display adapters is installed, what video mode the screen is currently in, which page of screen memory is being used, and the cursor position in all of the pages of screen memory (applicable with graphics cards only). Also, switch settings on the motherboard can be read via another ROM BIOS interrupt to determine what equipment is supposedly installed in the machine. Both the PCjr and EGA graphics boards also extend the basic graphic video modes supported. The standard IBM ROM BIOS supports both the monochrome and CGA adapters and 7 video modes using the CGA (Table 1). The EGA adapter extends the video_io ROM routines to support a number of other video modes that I won't go into this article.

| TEXT MODES | (columns by rows) | |
|---|---|---|
| Mode 0 | 40 x 25 | black and white (graphics adapter) |
| Mode 1 | 40 x 25 | color text (graphics adapter) |
| Mode 2 | 80 x 25 | black and white text |
| Mode 3 | 80 x 25 | color text (graphics adapter only) |
| GRAPHICS MODES | | |
| Mode 4 | 320 x 200 | pixels 4-color graphics |
| Mode 5 | 320 x 200 | pixels black and white (color burst off) |
| Mode 6 | 640 x 200 | pixels black and white |
| Mode = 7 | 80 x 25 | black and white text (monochrome adapter) |

**Table 1: BASIC VIDEO DISPLAY MODES**

One of the difficulties with other graphic adapter cards on the market is the lack of a standard way to determine that an adapter is installed. For example, there is no standard way to determine using software whether a Hercules monochrome graphics adapter (748 x 348 resolution) is installed. The same goes for the AT&T PC6300 with 640 x 400 resolution. For example, Microsoft WORD™ tries to test for the PC6300 by checking the ROM to locate a signature unique to this machine, which has to be patched each time a new PC6300 ROM is issued. For many of these other display cards, an install program has to be written to ask the user which display adapter is in the machine. And of course, plenty of naive users don't know diddly squat about the innards of their so-called "productivity tools."

Well, the memory layout of the CGA in the graphics mode is a bit painful to work with. The screen consists of 16K of dual ported memory mapped from B800:0 through B800:3FFFh. In the highest resolution graphics modes, rather than being a nice big linear map, the screen is mapped as two memory blocks. The even scan lines start at B800:0, and the odd scan lines start at B800:2000h. If you've done your arithmetic correctly, you might notice that there are 192 bytes of screen memory left unused at the ends of these two graphics memory blocks. Anyone got any swell ideas on how to use it?

Let's say you are in the black and white 640 x 200 pixel graphics mode (will someone tell me how to print a color graphics screen with only a black ribbon?). How should you map the IBM screen to the printer? If you measure the typical IBM screen display area, you will find that both height and width are about 6 inches. Well, golly, those screen dots aren't even close to being square or round (whichever your preference). This is known among techies as the aspect ratio and is what makes graphics such a challenge. Basically, without a 1 to 1 aspect ratio (a.k.a. that little Macintosh screen), there is no way to accurately map from a graphic image dump to a printer. One must know the *contents* of the image in order to do a proper translation (a circle, for example, which we will now draw as a circle on the printer).

The Okidata 92/93 printers support a rather crude 72 x 72 dots per inch for a total of 1152 horizontal dots on 16-inch-wide paper (612 dots on the 92). Rather than go to a lot of trouble, this code merely doubles the vertical dots when you use the 640 x 200 mode and doubles in both directions when you use 320 x 200 mode so that the image will be closer to the aspect ratio of the screen. The other little translation problem is that dot matrix printers impact a column of say 7–9 dots at one time while the IBM screen stores 8 dots (620 x 200 mode) in a horizontal line in each byte. Our software has to handle this translation from multiple screen memory bytes on various scan lines to one printer graphics byte as illustrated below:

```
                                *   Typical
Screen Byte                     *
                                *    9-pin
* * * * * * * *   ====>        *
                                *   Printhead
(multiple scan lines)           *
                                *
                                *
                                *
```

Continued . . .



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Software Tools

For Programmers & Non-Programmers

**Get 'State of the Art' performance and save valuable time with these high quality utilities!**

## Opt-Tech Sort™

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can even read the data. Example: 1,000 records of 80 bytes can be read, sorted and a new file written in less than 10 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or called as a subroutine to over 25 different programming languages.

All the sorting, record selection and reformatting facilities you need are included. A partial list of features includes: The ability to process files of any size. Numerous filetypes are supported including Sequential, Random, Delimited, Btrieve, dBASE II & III and many others. Up to 10 key fields can be specified (ascending or descending order). Over 16 different types of data supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes.

MS-DOS $149. ★ NEW ★ Xenix $249.

## ★ NEW ★ VERSION On-Line Help™ ★ NEW ★ VERSION

On-Line Help allows you to easily add "Help Windows" to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable Help Systems.

The resident version allows you to add help to any system. Your Help System is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire.

The callable version allows you to easily display help windows from your programs. A simple call to the help system makes the window appear. The original screen is automatically restored when the help window is cleared. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location.

MS-DOS $149. **Demo $10.** (apply toward purchase).

## Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit your previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. It's a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted.

To order or to receive additional information just call and receive immediate highly qualified attention!

**Opt-Tech Data Processing**

P.O. Box 678 — Zephyr Cove, NV 89448

(702) 588-3737

Number 16 on the Reader Service Card


**Finally, the Code**

I'll let the assembly language speak for itself, with a little brief explanation. The first thing we do after invocation is to jump to our initialization routine. During initialization, the program tests to make certain whether it has already been installed. This is easily done by comparing a string with the current INT 5 and our code. If already there, we exit and tell the user so. If not, we save the original INT5 vector, install our new vector, then terminate and stay resident. I chose to use the older method using INT 27 to stay resident at the time so as to support all DOS versions (otherwise you would have to test the DOS version before issuing the more recent stay resident DOS function).

Once installed, our new INT5 vector is invoked by the Shift-PrintScreen key. The general interrupt code sequence is as follows:

1. Check to see if PrintScreen is already executing or an error has occurred. In these cases just return.
2. Check the current video mode; if not one of the graphics modes, this coding just passes to the original PrintScreen vector.
3. Save the incoming stack just to make sure there's enough space.
4. Save the current video state (SAVE_STATE) in anticipation of doing something else with this basic code. We really need only a couple variables at most.
5. Save the current video screen (SAVE_SCREEN) to our own 16K buffer. This was done primarily in anticipation of putting a message on the screen to ask for confirmation or for future screen capture code. I have also unscrambled the video memory, so we have a nice linear bit map again for convenience. I really haven't been overly concerned about memory used by this TSR because with the screen buffer it would have to be at least 16.5K.
6. Print from our screen buffer (PSCREEN), doing the appropriate translation of crude aspect ratios and horizontal line to vertical column bytes.
7. During the print routine, we first initialize the printer to the graphics mode (GRAPH_INIT) and turn on graphics (GRAPH_ON). We then do the translation using mostly registers for speed although it hardly matters because in this application our code will run hundreds of times faster than any printer in the graphics mode. The translation uses bit testing and shifting of the mask byte to translate from lines to columns. At the end of each line, we send a graphics linefeed (GR_CRLF). And finally, after completing the print out, we restore the printer to some harmless default state (GRAPH_END).
8. If no errors occurred, set the status_byte flag in the ROM BIOS back to zero and return.

The specific printer routines and translation sequence are easily customized for another graphics board or printer. I could as easily have used code for the AT&T PC6300 (640 x 400 mode) and a Laserjet Plus for illustration. Now watch those dots and pixels dribbling by 'til we meet again. □

*Code follows.*




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Code from Steven Baker's "A Graphics Print Screen Utility..."

1

```
TITLE   'Graphics Print Screen Routine for MSDOS 4-1-85'

;
;       Graphics Print Screen Function for MSDOS
;       Version for Okidata 92/93
;       Author:         M. Steven Baker
;       Revision date   March 25, 1985
;       Last revision   April 1, 1985
;
;
;       Purpose:
;               installs and stays resident in DOS or CPM86 to revector
;               the print screen routine to allow a graphics dump
;
;       Operation:
;               This program must be run first to install a new
;               more powerful Print Screen Routine for PCDOS
;               which revectors the INT5 to this new code
;               This code stays resident until rebooting
;
;       E Q U A T E S
;
cr      equ     0dh
lf      equ     0ah
ESC     equ     1bh
;
false   equ     0
true    equ     not false
;
MASM    equ     true

        if      MASM            ;use some Macros
jmps    macro   dummy
        jmp     short   dummy
        endm
;
rs      macro   count
        db      count dup(?)
        endm

cseg    macro
        CODE    segment
        assume  cs:CODE,ds:CODE,es:CODE
        endm

        ENDIF   ;MASM


;       OKIDATA CHAR equates
;
ETX     equ     03      ;turn on graphics mode
STX     equ     02      ;exit graphics mode
;
pins    equ     6       ;number of pins on printer used per pass
; not set to seven, because of problems with doubling rows under
; 640 x 200 to maintain a similar aspect ratio and
; under 320 x 200 mode to maintain reasonable size
;
; 640 x 200 screen pel maps to printer 1 dot wide  x 2 dots high
; 320 x 200 screen pel maps to printer 2 dots wide x 2 dots high
;
;       ROM BIOS Data Area Equates
;
biosdata        equ     40h     ;segment of rom bios data
;
equip_flag      equ     10h     ;ROM bios data area installed hardware
crt_mode        equ     49h     ;current crt mode
crt_cols        equ     4ah     ;number of columns on screen
crt_len         equ     4ch     ;length of regen in bytes
crt_start       equ     4eh     ;starting address in regen buffer
cursor_posn     equ     50h     ;cursor position for each of 8 pages
;
cursor_mode     equ     60h     ;current cursor mode setting
```

2

```
active_page     equ     62h     ;current page being displayed
addr_6845       equ     63h     ;base address of active display card
crt_mode_set    equ     65h     ;current settings of the 3x8 register
crt_palette     equ     66h     ;current palette setting color card
;
status_byte     equ     100h    ; xxdata BIOS print screen flag
;
        cseg

        ORG     0100h
;
; Sign on message
;
start:  jmp     init            ;jump to initialization code
vernm   db      'Graphics Print Screen Version 1.0  3-28-85',0
;
newint5:
        pushf                   ;our interrupt 5 entry
        sti                     ;maybe a NOP later
        push    ds              ;save entry DSEG
        mov     cs:axsave,ax    ;and AX
;
        mov     ax,biosdata     ;setup segment pointer to 40h:
        mov     ds,ax
        test    byte ptr ds:status_byte,1 ;see if print in process or error
        jnz     exit
;
        mov     al,byte ptr ds:crt_mode ;get crt_mode
        mov     cs:vcrt_mode,al         ;and save it in our code
;
        cmp     al,4
        jb      no_graphic
        cmp     al,6
        ja      no_graphic
;
        mov     cs:oldss,ss             ;and save entry SSEG
        mov     cs:oldsp,sp             ;and stack Pointer
;
        mov     ax,cs                   ;restore AX=CS
        cli                             ;shut off interrupts
        mov     ss,ax                   ;now setup local stack
        mov     sp,offset stack
;
        call    graphic
;
        mov     ax,cs:axsave    ;restore AX
        cli                     ; for safety sake
        mov     ss,cs:oldss     ; restore SSEG and SP
        mov     sp,cs:oldsp
;
exit:   pop     ds              ;restore registers used and return
        popf
        iret

no_graphic:
        mov     ax,cs:axsave    ;restore AX
        pop     ds
        popf
        db      0eah
int5    dw      0,0             ; kluge jmpf
;

;
;       G R A P H I C
;       subroutine to do graphic dump to printer
;       Entry   DS = 40h ROM bios data area
;       all registers used should be preserved except AX,DS and Flags
graphic:
        mov     byte ptr ds:status_byte,1 ;say that we are doing a
                                          ;screen print
;
        sti                     ;re-enable interrupts
        push    bp
        push    bx
```


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        push    cx              ;save some registers
        push    dx
        push    si
        push    di
        push    es
        cld                     ;set forward direction
        call    save_state      ;save current video graphics state
        call    sav_scrn        ;save screen memory
        mov     ax,cs
        mov     ds,ax           ;set DS to our CS
;       call    ask             ;hooks to ask for print screen verify
        call    pscreen         ;go print screen
;
        mov     ax,biosdata
        mov     ds,ax                           ;point to ROM BIOS data area
        cmp     byte ptr ds:status_byte,0ffh    ;test for errors
        je      graphic2
        mov     byte ptr ds:status_byte,0       ;no longer doing print screen
graphic2:
        pop     es
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     bp
        ret

;       S A V E _ S T A T E
;       save the state of the current video
;       Entry   DS = 40h ROM Bios data area
;       EXit    ES,AX,CX,SI,DI are destroyed
;
 save_state:
        mov     ax,word ptr ds:equip_flag
        mov     cs:vequip_flag,ax       ;save it in our code
        test    ax,30h
        mov     ax,0b800h       ;segment for color card
        jne     sav_2
        mov     ah,0b0h         ;segment for black and white card
 sav_2: mov     cs:vidram,ax    ;store pointer to video ram
 ;
        mov     si,crt_cols             ;point to CRT_cols byte at seg 40h
        mov     di,offset vcrt_cols     ;and to data in our code
        mov     ax,cs
        mov     es,ax                   ;setup ES to our code
        mov     cx,(crt_palette-crt_mode)+1
        rep     movsb                   ;and move data
        ret

;       S A V _ S C R N
;       save the current video screen in our memory buffer
;       Entry   DS = 40h ROM Bios data area
;       EXit    DS,ES,AX,CX,SI,DI are destroyed
;
 sav_scrn:
        mov     ax,cs
        mov     ds,ax           ;set DS to our CS
        mov     es,ax           ;setup destination to our screen
        mov     di,offset scr_mem ;buffer at scr_mem
        mov     al,vcrt_mode    ;check current CRT mode
        cmp     al,4            ; is it 320 x 200 graphic
        jb      sav_scrn2
        cmp     al,6            ; or is it 640 x 200 graphic
        ja      sav_scrn2
;
;       save graphics screen in our screen memory buffer
;       reorganize as we move so that our buffer is sequential
;       ie odd lines follow even in our buffer not like the
;       crazy scheme used by the IBM graphics card with even scan
;       lines at b8000 and odd lines as ba000
;
```



```
sav_ginit:                      ;initialize for graphics move
;
        xor     ax,ax           ;zero ax
        mov     evenptr,ax      ;setup source even pointer to 0
        add     ax,2000h        ;setup odd pointer to 2000h
        mov     oddptr,ax
        mov     ax,0b800h       ;point to video ram
        cmp     ax,vidram       ;should be equal to DOS video ram
        je      sav_g1
;       call    errprt          ;hooks for an error routine
;       db      cr,lf
;       db      'DOS Video Ram Pointer does not agree with CRT_MODE'
;       db      cr,lf,0
;
sav_g1: mov     ds,ax           ;setup for DS register for move
        mov     cx,100          ;set up for 100 scan line pairs
;
evenline:
        push    cx              ;preserve CX
        mov     cx,40           ;40 words per line
        mov     si,cs:evenptr   ;get even scan line ptr
        rep     movsw
        mov     cs:evenptr,si   ;save SI pointer
oddline:
        mov     cx,40
        mov     si,cs:oddptr    ;get odd scan line pointer
        rep     movsw
        mov     cs:oddptr,si    ;and save SI pointer
;
        pop     cx
        loop    evenline        ;and repeat for number of line pairs
        ret
;
;       just move the block of video ram memory to our screen buffer
;       with no translation, for alpha modes of graphics card and
;       Monochrome video card (4k buffer)
;
sav_scrn2:                      ;not in graphics mode so don't translate
        mov     ax,vidram       ;pointer to video ram
        mov     ds,ax           ;setup source pointer to video ram
;
        xor     si,si           ;zero si
        mov     cx,4000h/2      ;move 2000h words (16k bytes)
        cmp     cs:vidram,0b800h        ;operating under graphics card
        je      sav_scrn3
        mov     cx,1000h/2      ;if not then move 800h words (4k bytes)
sav_scrn3:
        rep     movsw           ;and save screen
        ret
;
;
;       P S C R E E N
;       print screen routine for Okidata 92/93
;
pscreen:
        call    graph_init              ;initialize printer
        call    graph_on                ;turn on graphics_mode
        cmp     vcrt_mode,6
        je      pr640
;
pr320:                                  ;hook to handle color modes
                                        ; if desired
;
pr640:  mov     bx,offset scr_mem       ;setup pointer to buffer
;
        mov     cx,2*200/pins           ;number of graphic CHAR lines
doline: push    cx                      ;save outer loop counter 1 (lines)
        mov     cx,640/8                ;640 CHAR's per line
;
dobyte: push    cx                      ;save loop counter 2 (bytes)
        mov     dx,0180h                ;initialize test bit and mask
        mov     cx,8                    ;set to bit counter 8 pels/byte
        cmp     vcrt_mode,6             ;under 620x200
        je      tbit
;
```



PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        mov     dx,0184h                ;initialized for 320 x 200
        mov     cx,4                    ;4 pels per byte
;
tbit:   push    cx                      ;save loop counter 3 (bits)
        mov     cx,pins
        xor     ax,ax                   ;set AH=0 and AL=0
        xor     si,si                   ;zero si as our index
tbit_a:
        test    [bx+si],dl              ;now test with dl
        jz      tbit_b0
        or      al,dh                   ;OR in value of DH
;       cmp     vcrt_mode,6
;       jne     tbit_b                  ;if not 640x200 mode
;
; we now double bit for both modes
;
        shl     dh,1                    ;double bit for aspect ratio
        or      al,dh                   ;and or into AL
        dec     cx                      ;decrement count
        jmps    tbit_b                  ;and skip over nor ORing code
tbit_b0:
        shl     dh,1                    ;shift mask left
        dec     cx                      ; and decrement count for doubling

tbit_b:
        shl     dh,1                    ;double DH
        add     si,80                   ;add index to next scan line
        loop    tbit_a                  ;loop til first CHAR formed
;
        call    list_gr                 ;send CHAR to list device
        cmp     vcrt_mode,6             ;in 640 x 200 mode
        je      tbit_c
        call    list_gr                 ;if not, then send twice
;
tbit_c: pop     cx                      ;restore loop counter 3 (bits)
        shr     dl,1                    ;change test bit
;
        cmp     vcrt_mode,6             ;operating in 640 x200 mode
        je      tbit_d
;
        shr     dl,1                    ;shift mask again (2 bits=pel)
        dec     cx                      ;reduce bit count again
tbit_d: mov     dh,1                    ;re-init DH mask
        loop    tbit                    ;loop til all 8 bits done
;
        pop     cx                      ;restore loop counter 2 (bytes)
        inc     bx                      ;increment scr_mem pointer
        loop    dobyte                  ;do next byte
;
        call    gr_crlf                 ;send graphics CRLF
;
newscan:
;       cmp     vcrt_mode,6
;       je      newscan_2
;       add     bx,80*(pins-1)          ;point to next block of scan lines
;       jmps    newscan_3
newscan_2:
        add     bx,80*((pins/2)-1)      ;point to next block of scan lines
;
        pop     cx                      ;restore loop counter 1 (lines)
        loop    doline
;
        call    gr_crlf                 ;send graphics CRLF
        call    graph_off               ;turn off printer graphics mode
        call    graph_end               ;restore defaults to printer
        ret
```



```
;
;       L I S T _ G R
;       send CHAR to list device as graphic CHAR
;       test for ETX as char and doubles if so
;       Entry   AL= char
;       Exit    all registers are preserved
;
list_gr:
        cmp     al,ETX
        jne     list_gr2
        call    listout
list_gr2:
        call    listout
        ret
;
;       G R A P H _ I N I T
;       initialize printer to settings need by graphics
;       print routines
;       Entry   none
;       Exit    all registers preserved
;
graph_init:
        push    ax
        push    si
        mov     si,offset init_string   ;pointer to init string
        call    lstrng
        pop     si
        pop     ax
        ret
;
;       G R A P H _ E N D
;       initialize printer to normal settings on exit from pscreen
;       Entry   none
;       Exit    all registers preserved
;
graph_end:
        push    ax
        push    si
        mov     si,offset end_string    ;pointer to ending string
        call    lstrng
        pop     si
        pop     ax
        ret

;
;       G R _ C R L F
;       send text carriage return and linefeed while in
;       graphics mode; for OKIDATA send text crlf for 6/72"
;
gr_crlf:
        push    ax
        mov     al,ETX
        call    listout
        mov     al,lf
        call    listout
        pop     ax
        ret
;
;       G R A P H _ O N
;       turn on graphics printer mode
;
graph_on:
        push    ax
        mov     al,ETX
        call    listout
        pop     ax
        ret
;
graph_off:
        push    ax
        mov     al,ETX
        call    listout
        mov     al,STX
        call    listout
        pop     ax
        ret
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
;
;       P C R L F
;       send a normal CR, LF to printer
;       Entry   none
;       Exit    all registers preserved
;
pcrlf:  push    ax
        mov     al,cr
        call    listout
        mov     al,lf
        call    listout
        pop     ax
        ret
;
;       L S T R N G
;       send to list device the string pointed to by SI
;       entry   DS:SI = pointer to string
;                       with first byte equal to char count
;       exit    SI and AX destroyed
;
lstrng: push    cx              ;save CX for counter
        lodsb
        or      al,al           ;was it zero
        jz      lstrng_ret
        xor     cx,cx           ;xero count
        mov     cl,al
lstrng2:
        lodsb
        call    listout
        loop    lstrng2
lstrng_ret:
        pop     cx
        ret
;
;       L I S T O U T
;       send CHAR to current list output device
;       Entry   AL= char to print
;       Exit    all registers preserved assuming bdos only
;               destroys AX and BX; or ROM BIOS destroys AX
;
listout:
        push    ax      ;save CHAR
        push    cx
        push    dx      ;and register DX
        push    bp

        xor     dx,dx   ;set to printer 0
        xor     ah,ah   ;set AH =0 to print CHAR
        int     17h     ;and and print it
        test    ah,1    ;test for timeout error
        jnz     listerr
;
listout2:
        pop     bp
        pop     dx
        pop     cx
        pop     ax      ;restore AX
        ret
;
listerr:
        push    ds
        mov     ax,biosdata
        mov     ds,ax
        mov     byte ptr ds:status_byte,0ffh    ;set error flag
        pop     ds
;
        jmps    listout2
;
;       I N I T
;       initialization code for installing our interrupt
;
init:   mov     ax,cs           ;setup segment registers
        mov     ds,ax
        mov     es,ax
```



```
        pushf                   ;save flags
        pop     bx
        cli
        mov     ss,ax
        mov     sp,offset stack
        push    bx
        popf                    ;restore flags
;
;       mov     dx,offset signon
;       call    pmsg            ; say HI
;       CALL    crlf
;
; copy print screen interrupt vector to our JUMP FAR return
; used to return from our checking code
;
movint: xor     ax,ax           ;zero AX
        mov     ds,ax           ;set DS to segment 0
        cld                     ;set forward direction
        mov     si,5*4          ;pointer to int5
        mov     di,offset int5  ;pointer to our JUMP FAR code
        mov     ax,cs
        mov     es,ax           ;set ES to our code segment
        movsw                   ;move offset
        movsw                   ;move segment
;
; now test that we have not already previously installed this
; our graphic print screen interrupt
;
testint:
        mov     si,14h          ;pointer to int5
        lodsw                   ;get offset to AX
        mov     dx,ax           ;save offset to DX
        lodsw                   ;get segment to AX
        mov     es,ax           ;temporrarily store it in ES
        mov     ax,cs           ;restore our DS register
        mov     ds,ax
        mov     ax,offset newint5
        cmp     ax,dx           ;offsets are not equal
        jne     setint
;
        mov     si,offset vernm ;point to version name
        mov     di,si           ;in both SI and DI
        mov     cx,(offset newint5)-(offset vernm)
testint2:
        cmpsb                   ;compare them for equal
        jne     setint
        loop    testint2
        mov     si,offset dupmsg
        call    pmess
        jmp     done_2
;
;
; now replace INT5 interrupt vector with pointer to our code
;
setint: mov     ax,cs
        mov     ds,ax   ;setup DS = CS
        mov     bx,ax   ;save our code segment in BX
        xor     ax,ax   ;zero AX
        mov     es,ax   ;setup ES as segment 0
        mov     di,14h  ;point to INT5 vector
;
        cld             ;set forward direction
        pushf
        cli             ;shut hardware interrupts off
;
        mov     ax offset newint5       ;setup our int5
        stosw
        mov     ax,bx   ;now set our code segment
        stosw           ;and store it
;
inton:  popf            ;turn back on interrupts
                        ;vector has now been replaced
;
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        mov     ax,cs
        mov     ds,ax
        mov     si,offset endmsg        ;tell them w'ere done
        call    pmess
;
;
done:   mov     dx,offset intend+1
        int     27h             ; terminate but stay resident under dos
;
; exit but don't stay resident
;
done_2:
        mov     ax,0
        mov     dx,ax           ;set DX =0 for CPM86 exit
        jmp     bdos

;
;
; CP/M interface
;
crlf:   MOV     AL,cr
        CALL    pchar
        MOV     AL,lf
;
pchar:  PUSH    DX
        PUSH    BX
        MOV     DL,AL
        MOV     ah,2
        call    bdos
        POP     BX
        POP     DX
        RET
;
;       P M E S S
;       print message to screen
;       Entry   SI = pointer to message terminated by null byte
;       Exit    AL and SI destroyed
pmess:  lodsb                   ;get byte
        or      al,al           ; is it zero??
        jnz     pmess2
        ret
pmess2: push    si              ;save pointer
        call    pchar           ;send character
        pop     si              ;restore pointer
        jmps    pmess           ; and continue

bdos:   int     21h
        ret
;
; message texts
;
endmsg  db      'Graphics Print Screen Code has been installed'
        db      cr,lf,0
;
dupmsg  db      'Graphics Code was already there ..exiting'
        db      cr,lf,0
;
;special control strings for Okidata 92/92

init_string     db      8               ;byte count to send
                db      18h             ;clear buffer and reset
                db      1ch             ;12 CPI
                db      cr,lf           ;send at least 1 CRLF
                db      ESC,'%9'        ;set line spacing
                db      pins*2          ; to n/144"
                db      0,0,0           ;extra space
;
end_string      db      7
                db      18h             ;clear buffer and reset
                db      1ch             ;12 CPI
                db      ESC,'6'         ;designates 6 Lines per Inch (LPI)
                db      cr,lf,lf        ;send  CRLF
                db      0,0,0,0         ;extra space
;
```



```
;
;       Video Display Data Area
;

vequip_flag     dw      0       ;ROM bios data area installed hardware
;
vcrt_mode       db      0       ;current crt mode
vcrt_cols       dw      0       ;number of columns on screen
vcrt_len        dw      0       ;length of regen in bytes
vcrt_start      dw      0       ;starting address in regen buffer
vcursor_posn    dw      0,0,0,0 ;cursor position for each of 8 pages
                dw      0,0,0,0
;
vcursor_mode    dw      0       ;current cursor mode setting
vactive_page    db      0       ;current page being displayed
vaddr_6845      dw      0       ;base address of active display card
vcrt_mode_set   db      0       ;current settings of the 3x8 register
vcrt_palette    db      0       ;current palette setting color card
                dw      0
;
vidram          dw      0       ;pointer to active video ram
evenptr         dw      0       ;pointer to even lines
oddptr          dw      0       ;pointer to odd lines
;
axsave          dw      0       ;storage for AX on entry
oldss           dw      0       ;storage for entry SSEG
oldsp           dw      0       ;storage for old stack Pointer
;
        RS      100h
stack   dw      0
        dw      0
;
scr_mem equ     $               ;buffer for 16k of screen memory
intend  equ     $+4000h

        CODE    ends
        end     start           ;put starting label routine at 100h
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Coding Guidelines for Resident Programs

by
**Graham Pearson**

*This paper was presented at the Houston HAL PC Assembler Special Interest Group Meeting on November 25, 1986. It follows up M. Steven Baker's article "DOS: Terminate and Stay Resident,"* Programmer's Journal, *November/December 1986, pages 26–30.*

## Introduction

Steven Baker's excellent article provides a thorough history of DOS Terminate and Stay Resident (TSR) programs and the special features and DOS functions and interrupts they should incorporate.

This article illustrates how these features are implemented by providing a short but heavily commented example. This example illustrates other features of TSRs outside the scope of Steven's article, such as how a TSR can be removed from memory so that all space is returned to DOS and how to stop a TSR from "fluttering" (rapidly activating and deactivating) if the hot keys are held down too long.

## Example Overview

The example at the end of this article doesn't perform any useful function. Its purpose is to illustrate how a TSR should be installed or deinstalled and under what circumstances the body of a TSR should be given control.

The TSR code can be divided into three parts:

1. a jmp instruction, which transfers control to the TSR install code
2. the TSR body (the code to be made resident)
3. the TSR install code

With this arrangement of code, memory requirements are minimized. Once a TSR has been installed, there is no need to reserve the area occupied by the install/deinstall code. The TSR body doesn't need it, so DOS can reuse this area for other purposes. When terminating, the install code establishes a new DOS "high water mark" at the end of the TSR body.

## Installing a TSR

DOS function 35H should be used to retrieve the current interrupt vectors, and function 25H should be used to update these vectors. Whenever interrupts are redirected, your program should always "chain" to the original vector, preferably before you process the interrupt. By adopting this "me last" attitude, you are being a good neighbor. This strategy goes a long way to ensure problem-free cohabitation of several resident programs in memory at one time.

DOS function 31H should be used as the terminate and stay resident mechanism. This has two advantages over the older interrupt 27H: It allows a completion code to be returned to the "parent" process (DOS), and resident programs needing more than 64K of memory can be installed using this function.

DOS 1 versions don't provide the functions recommended in the two paragraphs above. Because of this, the install process should first make sure that DOS 2.00 or later is being used. If DOS 1 is being used, the installation process should be aborted using interrupt 20H as illustrated in the example.

A "handle" should be installed in some safe memory location. This handle occupies four bytes, the same area as that occupied by a single interrupt vector. The handle has two components and two purposes. First, a two-byte segment value enables your resident program to be deinstalled as described in the following section. Second, a two-byte signature prevents your program from being loaded into memory more than once by an absent-minded user.

Microsoft has designated interrupt vectors 60H through 7FH as available for user definition. One of these can be used as a safe memory location for your handle. Rather than reserve a specific vector, your program should search this area for an available vector. The example illustrates this process.

Let's suppose your resident program uses a variety of DOS functions to display a disk directory. A user has loaded your utility into memory and is now running a database application (foreground process) that also uses a variety of DOS functions for file I/O. If your utility is ever activated while DOS is performing some file I/O for the database, it's reboot time! It's also the last time your user will ever use your utility because the odds are that the system lock-up will corrupt the database files.

Many DOS functions are not re-entrant because they utilize one or more internal stack areas. If a DOS function is interrupted and another DOS function is reinvoked by the interrupter, the internal stack values being used by the first DOS invocation are corrupted by the second.

DOS functions 1 through C can be safely interrupted at any time. However, these functions provide only low-level tasks such as keyboard input and screen output. How can a resident program safely use the high-level DOS functions such as file I/O? The answer lies in the undocumented DOS function 34H, discussed in detail in Steven Baker's article.

DOS function 34H returns the address of a one-byte flag that indicates whether or not DOS is performing a non re-entrant process. A non-zero flag value indicates that DOS must not be interrupted. The flag address is returned in ES:BX. The install section of the example illustrates how function 34H is used. The section handling interrupt 9 illustrates how this flag is used to decide whether or not the TSR body should be given control.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Many programmers have serious reservations about using undocumented functions. DOS function 34H has remained consistent since Version 2.00. You must decide whether or not the risk of using function 34H is outweighed by the benefits it provides.

Technically-oriented users appreciate knowing how much RAM they are sacrificing by installing your program. Because the resident program size is calculated as a prerequisite to the DOS terminate and stay resident function 31H, this information appears to be at your fingertips.

However, there are four areas of memory associated with each program loaded into memory by DOS:

1. the "obvious" area—your TSR code, including 100H bytes of program segment prefix (PSP) created by DOS during the loading process
2. a 16-byte DOS memory control block (MCB) occupying the paragraph of memory immediately below your PSP
3. a variable-sized DOS environment area (containing the COMSPEC=, PATH=, and PROMPT= strings, etc.)
4. a 16-byte DOS memory control block (MCB) occupying the paragraph of memory immediately below this DOS environment area.

The only unknown in the list above is the size of the DOS environment area. This can be retrieved by the following process:

1. Retrieve the segment address of the environment from the word at offset 2CH in your program's PSP.
2. Calculate the segment address of its DOS MCB by decrementing this value by one.
3. Retrieve the size of the DOS environment area from the word at offset 3 in this DOS MCB.

The following assembler code accomplishes this sequence. The register BX finishes up with the DOS environment size in paragraphs (1 paragraph = 16 bytes).

```
push    ds              ;save DS for later restore
mov     bx,ds:[2CH]     ;get environment address
dec     bx              ;point DS to the
mov     ds,bx           ;environment's DOS MCB
mov     bx,ds:[3]       ;retrieve environment size
pop     ds              ;restore DS
```

The total of all four areas listed above corresponds to the reduction in "bytes free" reflected when the DOS CHKDSK command is run before and after installing a TSR.

### ROM BIOS Diskette I/O

In his article, Steven Baker also recommends that interrupt 13H should be monitored for activity as it is not re-entrant. This would involve redirecting interrupt 13H during the install process, restoring it during the deinstall process, and including the following code as an interrupt 13H replacement:

```
diverted_int13:
        pushf                           ;save the flag register
        mov             cs:int13_flag,0FFFFH  ;set out the program flag
        call_int13      db      09AH          that the diskette-interrup
        int_13_vect     label   dword     is in process
        int_13_off      dw      ?
        int_13_seg      dw      ?
        mov             cs:int13_flag,0       ;reset out diskette flag
        iret
```

and making sure the int13_flag is zero before giving the TSR body control. The example omits these steps for the sake of brevity. However, there are very few application programs that use BIOS interrupt 13H calls rather than DOS functions for their disk I/O.

### The TSR Body

Entry to the TSR body in the example is via either interrupt 9 or 28H. Interrupt 9 is appropriate if your TSR is to be triggered by a "hot key" combination. Interrupt 1CH is appropriate if your TSR is to be triggered by the timer tick. Interrupt 28H is described in Steven Baker's article. If entry is via interrupt 9 or 1CH, the dos_busy flag must be checked to make sure that DOS is not currently executing a non re-entrant function. The TSR body is given control only if the following two additional prerequisites are met.

The keyboard status word at offset 17H in the BIOS data segment (segment at 40H) is checked for ON bits matching the ON bits in the variable hot_keys. This check is appropriate only if interrupt 9 is being used for TSR entry, and it indicates whether or not the hot keys are being held down.

The timer count word at 6CH in the BIOS data segment is compared to that of the last time the TSR body was given control. A delay specified by DELAY_COUNT of 36 timer ticks, or about 2 seconds, is used in this example. This prevents the TSR body from "fluttering,." Fluttering takes place at 15 times



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Introducing

# Quaid Analyzer

## the tool that created CopyWrite

**Now you can debug your own programs with a professional quality debugger - the one that unraveled every form of copy-protection used on the PC.**

With the Quaid Analyzer, you can:

- ☐ See occurences of any interupt, with its meaning shown on the screen.
- ☐ View memory as text or instructions, scrolling as easily as you do with an editor.
- ☐ Run until a memory location or I/O port is changed.
- ☐ Protect your hard disk from accidental destruction.
- ☐ Analyze software without the source, even when it uses countermeasures to thwart tracing.
- ☐ See all stages of the boot load.

**We kept the Quaid Analyzer off the market to avoid helping publishers with copy-protection. Now that copy-protection is gone, we can sell it to you.**

The Quaid Analyzer is a software tool occupying 100K bytes. It runs on any IBM PC and most MS-DOS systems without hardware modification.

To order Call **(416) 961-8243**

**Quaid Analyzer $99 U.S.**

All orders shipped at our expense within a day. All major credit cards accepted.

or return coupon to:
45 Charles St. East
Third Floor, Dept. 603
Toronto, Ontario. M4Y 1S2

Payment method MC-Visa-Amex-Diners-Check

Card No. _____

Expiry Date _____

Name _____

Address _____

City/State _____

Phone No. _____

Signature _____

Quaid Software Limited

Ask about Disk Explorer the program that takes over where Quaid Analyzer leaves off.

Number 101 on the Reader Service Card

per second if a heavy-handed user holds the hot keys down until they auto-repeat. This phenomenon is typically accompanied by hot flashes both on and off the screen and is best avoided!

By using a variable for the hot keys bits rather than a fixed value to check for hot key activity, you make it easier for a configurator (a program other than the TSR program) to change the hot key combination to suit the user. In the example, the hot key byte is the fourth byte in the TSR program file (the jmp install instruction occupies the first 3 bytes).

To avoid the wrath of a careless user, the TSR body must make every attempt to avoid system lockup. This involves monitoring the DOS critical error interrupt 24H, especially if sensitive DOS functions such as file or printer I/O are invoked. Your users *will* open their floppy drive doors at the wrong time! Interrupt 24H must not be redirected by the TSR install process. Your interrupt 24H replacement code is appropriate only while your TSR is in control. Interrupt 24H must be redirected at the start of the TSR body and must be restored before control is returned to the application program. The example illustrates this coding convention.

### The Bottom Line

This article hopes to provide a useful followup to Steven Baker's excellent review of Terminate and Stay Resident programming. If you follow the coding guidelines illustrated by the example accompanying this article, your TSR body code should be able to use the full range of BIOS and DOS functions without fear of reprisal. ☐

*Code follows.*

## MEMORY RESIDENCY MADE EASY

Jack™ The Resident Program Developer's Kit, contains everything you need to create your own RAM resident software without the headaches of memory management, windows or DOS re-entrancy. What makes Jack™ unique is NO hooks and NO special code, just link Jack™ with your program. That's all there is to it. Now you can concentrate on how your program should run instead of worrying about how to make it memory resident.

Jack™ FEATURES:

- Virtually any C program can be made memory resident without any modification to the existing source code. Screens are automatically saved to conventional memory [or the expanded memory of an Intel Above Board] when popping up and restored upon popping down.
- DOS function calls and disk I/O can safely be used. Jack™ does not use any undocumented DOS calls to solve the DOS re-entrancy problem. [DOS 2.0 or greater required]
- Peaceful coexistence with other popular pop-up utilities.
- An easy to use windowing system, with source code, to minimize development time while maximizing flexibility.

Finally, a standard for easily creating memory resident software has arrived!

**To Order:** Please remit $199.95 [US] + S/H for Jack™ by VISA, written P.O., certified check or international money order to:

CrackerJack
Microsoftware Corporation.
200 Bay Street PO Box 86,
Toronto, Ontario. M5J-2J2. Canada.
(416) 865-9621.
Please specify either the Datalight C 2, Lattice C 3.0 and 3.1, or MicroSoft 4.0 version of Jack™.

Attention Software Developers, let's work together, please contact us regarding our distribution agreement.

Jack™ and CRACKERJack are registered trademarks of CrackerJack Microsoftware Corporation, Datalight is a registered trademark of Datalight Inc., Lattice is a registered trademark of Lattice Inc., Intel and ABOVE BOARD are registered trademarks of Intel Corporation, Microsoft and DOS are registered trademarks of Microsoft Corporation.

Number 88 on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Code from Graham Pearson's " Coding Guidelines..."

1

```
; This code illustrates how to install a TSR and the circumstances under
; which the body of the TSR should be given control.
;
; The hot keys in this example are the Alt + Left Shift keys.  This hot key
; combination is defined in the variable hot_keys and can be changed to suit
; the user by changing the bit pattern.
;
DELAY_COUNT     EQU     36              ;36 ticks = approx. 2 seconds

; Locations of BIOS Data needed by the resident program

bios_data       segment at 40H
        org             17H             ;Keyboard status flags
        kbd_status      dw      ?
        org             6CH             ;Timer count - low word. This rolls
        low_timer       dw      ?       ;around 0FFFFH once per hour
bios_data       ends


code            segment
        assume  cs:code, ds:nothing, es:nothing, ss:nothing
        org     100H                    ;Define starting point for .COM file
entry_point:
        jmp     install                 ;Skip over the code to be made resident

        hot_keys        dw      01010B  ;Hot key bits -> Alt + Left Shift
        this_time       dw      ?       ;Most-recent time hot keys detected
        trig_time       dw      ?       ;Last time resident code was triggered
        dos_busy        label   dword   ;Pointer to "DOS is busy" flag
        dos_busy_off    dw      ?       ;These values are initialized during
        dos_busy_seg    dw      ?       ;the "install" process
        criterr_flag    db      ?       ;Used by diverted INT 24H


; Replacement for INT 24H - critical DOS error
diverted_int24:
        mov     cs:criterr_flag,1       ;Switch error flag ON
        xor     al,al                   ;Tell DOS to ignore the error
        iret                            ;Return to DOS

        int_24_vect     label   dword
        int_24_off      dw      ?       ;These values are initialized each time
        int_24_seg      dw      ?       ;the TSR body takes control


; Replacement for INT 9 - keyboard hardware interrupt
diverted_int9:
        pushf                           ;Simulate an INT 9 to pass through the
        call_int9       db      09AH    ;original BIOS interrupt handler
        int_9_vect      label   dword
        int_9_off       dw      ?       ;These values are initialized during
        int_9_seg       dw      ?       ;the "install" process

        push    ds                      ;Save all registers used by this
        push    bx                      ;section of code
        lds     bx,dos_busy             ;See if DOS is busy
        cmp     byte ptr [bx],0
        pop     bx                      ;Restore all registers
        pop     ds
        jz      get_bios_data           ;Zero means DOS is not busy
dos_is_busy:
        iret                            ;Return control to interrupted process


; Replacement for INT 28 - generated by DOS, esp. during keyboard I/O functions
diverted_int28:
        pushf                           ;Simulate an INT 28H to pass through
        call_int28      db      09AH    ;the original DOS interrupt handler
        int_28_vect     label   dword
        int_28_off      dw      ?       ;These values are initialized during
        int_28_seg      dw      ?       ;the "install" process
```

2

```
get_bios_data:
        sti                             ;Restore interrupts
        push    ds                      ;Save all registers used by this
        push    ax                      ;section of code
        mov     ax,bios_data            ;Retrieve BIOS Data values
        mov     ds,ax
        assume  ds:bios_data
        mov     ax,low_timer            ;Retrieve and save timer count
        mov     cs:this_time,ax
        mov     ax,kbd_status           ;Retrieve keyboard status bits
        push    cs                      ;Point DS to our code segment
        pop     ds
        assume  ds:code
chk_keys:
        and     ax,hot_keys             ;See if hot keys are pressed
        cmp     ax,hot_keys
        jne     back_to_applic
chk_timer:
        mov     ax,this_time
        cmp     ax,trig_time            ;Enter resident routine if timer
        jb      time_is_right           ;count has rolled around the hour
        sub     ax,trig_time
        sub     ax,DELAY_COUNT          ;Transfer to resident code only if
        jnc     time_is_right           ;elapsed time between hot keys is
                                        ;greater than delay count
back_to_applic:
        pop     ax                      ;Restore all registers
        pop     ds
        iret                            ;Return control to interrupted process


time_is_right:
        mov     ax,this_time            ;Update latest trigger time
        mov     trig_time,ax
        pop     ax                      ;Restore all registers
        pop     ds

;*****************************************************************************

; This is the start of the application-dependent resident code

start_program:

        push    ax                      ;Save all registers used by the
        push    bx                      ;resident application
        push    cx
        push    dx
        push    si
        push    di
        push    bp
        push    ds
        push    es

        push    cs                      ;Point DS to our CS
        pop     ds

        mov     ax,3524H                ;Get current INT 24H vector
        int     21H
        mov     int_24_off,bx
        mov     int_24_seg,es
        mov     ax,2524H                ;Redirect INT 24H
        mov     dx,offset diverted_int24
        int     21H

;*****************************************************************************

        ; Include application specific code here

;*****************************************************************************

restore_int24:
        lds     dx,int_24_vect          ;Restore INT 24H
        mov     ax,2509H
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        int     21H
        pop     es                      ;Restore all registers
        pop     ds
        pop     bp
        pop     di
        pop     si
        pop     dx
        pop     cx
        pop     bx
        pop     ax
        iret                            ;Return control to interrupted process

; This is the end of the application-dependent resident code

        end_of_res      label   word

;******************************************************************************

; This procedure installs the resident section of code in memory.
; It performs the following tasks:
;
;    1  Make sure the operating system is at least DOS 2.00
;    2  Make sure the resident code is not already installed in memory
;    3  Retrieve an available vector for installing the signature
;    4  Retrieve vector to "DOS is busy" and save it for use by resident code
;    5  Update the interrupt vectors required by resident code.
;    6  Install the signature in the available vector found in step 3
;    7  Terminate and stay resident - transfer control back to DOS

install:
; Make sure that the operating system is at least DOS 2.00
        mov     ah,30H
        int     21H
        or      al,al
        jnz     chk_vectors
        mov     dx,offset baddos_msg    ;DS:DX -> Display "bad DOS" message
        mov     ah,9
        int     21H
        int     20H                     ;Return to DOS the old-fashioned way

; See if the resident code is already installed in memory.
; If not, retain an available interrupt vector for later use as a signature.

chk_vectors:
        push    ds                      ;Save DS -> CS
        xor     ax,ax                   ;Search 32 interrupt vectors starting
        mov     ds,ax                   ;at INT 60H - user interrupt area
        mov     si,60H * 4
        mov     cx,20H
next_vector:
        mov     ax,[si + 2]
        or      ax,ax                   ;If segment value is zero, get offset
        jz      get_vect_off            ;value
        cmp     ax,0D0C7H               ;Check interrupt segment value against
        jne     vector_loop             ;resident code signature
no_install:
        pop     ds                       ;Restore DS -> CS
        mov     dx,offset no_install_msg ;DS:DX -> Display "no install" message
        mov     ah,9
        int     21H
        mov     ax,4C01H                ;Return error code of 1
        int     21H                     ;Return to DOS with error code
get_vect_off:
        cmp     word ptr [si],0         ;If offset value is zero, this vector
        je      found_vector            ;is available for our use
vector_loop:
        add     si,4                    ;Interrogate next vector
        loop    next_vector
found_vector:
        ;Make sure we successfully complete the install process before
        ;installing the signature in this available interrupt vector
        pop     ds                      ;Restore Data Segment
        mov     free_vector,si          ;Save free vector for later
```

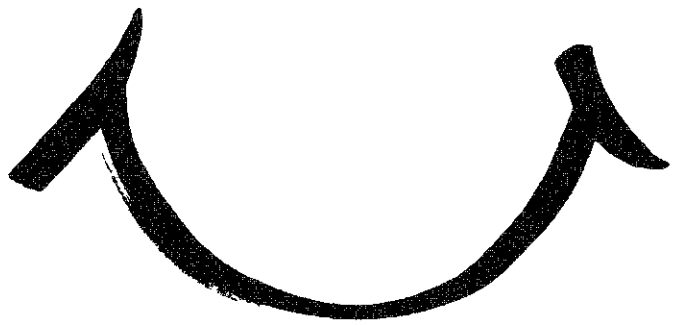


```
; Retrieve original interrupt vectors and install these in the resident code.
; Redirect the interrupt vectors to point to resident code.
; Install signature in available interrupt vector.

        push    es                      ;Save ES -> CS
        mov     ah,34H                  ;Retrieve segment + offset
        int     21H                     ;pointer to "DOS is busy" flag
        mov     dos_busy_off,bx
        mov     dos_busy_seg,es
        mov     ax,3509H                ;Get current INT 9H vector
        int     21H
        mov     int_9_off,bx
        mov     int_9_seg,es
        mov     ax,2509H                ;Redirect INT 9H
        mov     dx,offset diverted_int9
        int     21H
        mov     ax,3528H                ;Get current INT 28H vector
        int     21H
        mov     int_28_off,bx
        mov     int_28_seg,es
        mov     ax,2528H                ;Redirect INT 28H
        mov     dx,offset diverted_int28
        int     21H
        pop     es                      ;Restore ES -> CS
        mov     di,free_vector          ;Retrieve value found earlier
        push    ds                      ;Save DS -> CS
        xor     ax,ax                   ;DS -> interrupt vectors
        mov     ds,ax
        mov     [di],cs                 ;Install signature in the available
        mov     word ptr [di+2],0D0C7H  ;interrupt vector
        pop     ds                      ;Restore DS -> CS
; Display installed message
        mov     dx,offset success_msg   ;DS:DX -> Display "success" message
        mov     ah,9
        int     21H
; Prepare for program "terminate and stay resident"
        mov     dx,end_of_res           ;Calculate the amount of memory used
        dec     dx                      ;by the resident code
        or      dx,0FH
        inc     dx                      ;Round up to the nearest paragraph
        mov     cl,4                    ;Convert code size to paragraphs
        shr     dx,cl
        mov     ax,03100H               ;Return error code of 0
        int     21H                     ;Terminate but stay resident

        free_vector     dw      ?       ;Pointer to available interrupt vector
        copyright       db      '  Copyright (c) 1986 Kudos Software  '
        baddos_msg      db      'TSR will not run with DOS 1.XX'
                        db      0DH, 0AH, '$'
    no_install_msg      db      'TSR is already resident in memory'
                        db      0DH, 0AH,'$'
        success_msg     db      'TSR successfully installed in memory'
                        db      0DH, 0AH, '$'

code            ends
                end     entry_point     ;Define entry point for .COM file
```



PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# COMPUTER FAIRES® GIVE YOU AN UNFAIR ADVANTAGE.



The Northeast and West Coast Computer Faires are going to help you surge ahead into new business opportunities in 1987. Attendees will find the latest and most innovative products and services to put them ahead in the quest for productivity. Exhibitors searching for a sophisticated audience that's ready to buy will be smiling at their extra sales.

And extensive promotion, a full conference series, and special features like State-of-the-Art Professional Development Seminars help add up to give you the Computer Faire advantage—a smile of triumph.

**Attention exhibitors! Call (617) 449-6600, Ext. 4014, now to reserve exhibit space or to receive more information.**

**West Coast Computer Faire '87**
March 26–29, 1987, Moscone Center, San Francisco

**Northeast Computer Faire '87**
October 15–17, 1987, World Trade Center, Boston

**I want the Computer Faire advantage.**
**Send me ☐ exhibitor ☐ attendee information on ☐ Northeast ☐ West Coast Computer Faires**

Name _____ Title _____
Company _____
Address _____
City _____ State _____ Zip _____
Telephone ( _____ ) _____ PJ387

Return to the Interface Group, Inc., 300 First Avenue, Needham, MA 02194

Number **58** on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Parallel Processing with the EGA

by
**Michael Abrash**

This issue's title refers to the ability of the powerful EGA chip set to manipulate up to four bytes of display memory at once. The EGA provides four ALUs (Arithmetic Logic Units) to assist the CPU during display memory writes, and this hardware is a tremendous resource in the task of manipulating the EGA's massive bit map. The ALUs are actually only one part of the surprisingly complex data flow architecture of the EGA, but since they're involved in almost all memory access operations, they're a good place to begin.

There's more to cover, too–starting with this article, I'm going to briefly discuss interesting graphics hardware and software I've come across recently. Since most *PJ* readers are developers, I'm not going to be doing traditional reviews, but rather discussing products' strengths and weaknesses from the perspective of a developer who's deciding what products to support and/or develop for. This issue, rather than examine specific products, I'm going to report on the state of PC graphics, as observed at Fall Comdex.

## Comdex

The overriding graphics-related impression from Comdex was that the EGA is the standard for PC graphics. Virtually no one showed a CGA-level product, and just about all demos were run on EGAs or more powerful boards. One AT-compatible portable was shown: It had an EGA built into the motherboard. Many more built-in EGAs will surely follow, especially given the one- and two-chip EGA chip sets hitting the market. In short, imperfect as it may be, the EGA is the current standard for mass-market PC graphics. (On a related note, inexpensive ATs abounded at Comdex. The widespread use of ATs will help greatly in overcoming the relatively sluggish performance of the EGA.)

Given that the EGA is today's standard, the big question is clearly, "What's next?" One answer seems to be what many people are calling the EEGA, which is an EGA with higher-resolution modes than the IBM board's, typically 640 x 480 but ranging in some cases as high as 800 pixels in width and 500 pixels in height. Several EEGA boards were shown at Comdex, and others have already been announced. There's nothing revolutionary about the EEGA; it's a natural outgrowth of the development of EGA chip sets that can run faster than IBM's. EEGA hi-res modes are identical to the normal EGA 640 x 350 mode, with 16 of 64 colors available, except that the dot clock speed is higher and the portion of display memory used for video data at any given time is larger. (In 640 x 480 mode, for example, the active bit map size is over 37K; by contrast, 640 x 350 mode requires less than 28K.)

EEGAs are significant for several reasons. First, they provide enough resolution for emulation of many popular graphics terminals. Second, the market is being flooded with monitors that support the higher resolutions of these boards, such as the NEC MultiSync and Sony MultiScan. Third, 640 x 480 provides about a 1-to-1 aspect ratio; a 1-to-1 aspect ratio is one reason why Macintosh graphics look so good. Finally, support of EEGAs requires very little new programming since all the registers and memory addresses that are used in standard EGA programming are used by EEGAs as well, so software should become available for them quickly.

EEGAs are hardly a giant step forward, but they are a very practical move toward high-resolution graphics. The bad news is this: With a 30% larger bit map to manipulate, software running on the EEGA will be slower than EGA software. Among other implications, this makes it imperative that EGA software developers write the highest-performance code possible.

Not all the graphics at Comdex were EGA-related, though, by a long shot. There were a number of IBM Professional Graphics Controller (PGC) compatible systems (in fact, Zenith was demonstrating its 80386-based system with a Turbo PGC, which was interesting since virtually none of the speed of the graphics was attributable to the 386). There were systems built around both the Intel 82786 and the TI 34010 (which I'm planning to look at soon), and there were proprietary systems with resolutions up to 1K-by-1K. Unfortunately, there were no signs of standardization for graphics above the EEGA level. I suppose standardization will have to wait for either the TI or the Intel chip to become a mass-market success. (Or some other chip—it will happen eventually). Another possibility is that either or both chips may prove to be powerful enough to support good performance through device drivers, in which case we may (finally!) be near the era of true device independence.

IBM could, of course, set the next graphics standard with its rumored Turbo EGA, or with another graphics adapter. However, it doesn't seem to be as easy as it used to be for IBM to set standards; the PGC was hardly an unqualified success, and the EGA became a standard only because third party manufacturers imitated the technology and brought the price way down. It's not at all clear that if IBM were to come out with a new adapter tomorrow it would have much impact, particularly if the technology were sufficiently proprietary that other manufacturers couldn't clone it. For the near future, at least, fragmentation seems to be the nature of the high end of the PC graphics market.

On the bright side, many of the high-end display adapters featured EGA compatibility. Whatever direction the graphics



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

market takes in the next few years, it's safe to say that EGA software will retain its value, particularly if that software can support 640 x 480 graphics.

**EGA Programming: ALUs and Latches**

I'm going to begin our detailed tour of the EGA at the heart of the flow of data through the EGA: the four ALUs built into the EGA's Graphics Data Controller (GDC) chips. The ALUs (one for each display memory plane) are capable of ORing, ANDing, and XORing CPU data and display memory data together, as well as masking off some or all of the bits in the data from affecting the final result. All the ALUs perform the same logical operation at any given time, but each ALU operates on a different display memory byte. (Recall that the EGA has four display memory planes, with one byte in each plane at any given display memory address. All four display memory bytes operated on are read from the same address, but each ALU operates on a byte that was read from a different plane and writes the result to that plane.) This arrangement allows four display memory bytes to be modified by a single CPU write (which must normally be preceded by a single CPU read, as we will see). The benefits are clear: If the CPU had to select each of the four planes in turn via OUTs and perform the four logical operations itself, EGA performance would be slowed to a crawl.

Figure 1 is a simplified depiction of data flow around the ALUs. Each ALU has a matching latch that holds the byte read from the corresponding plane during the last CPU read from display memory—even if that particular plane wasn't the plane the CPU actually read on the last read access. (Only one byte can be read by the CPU with a single display memory read; the plane supplying the byte is selected by the Read Map register. However, the same address from all four planes is always read when the CPU reads display memory, and those four bytes are held in their respective latches.)

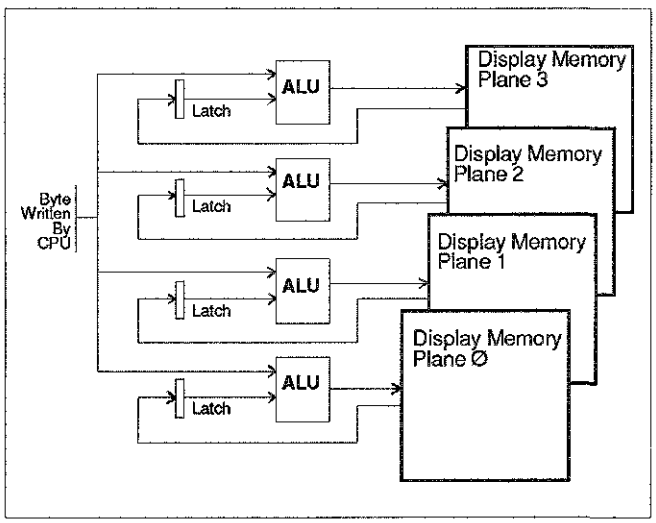



**Figure 1:** Simplified Data Flow Around the Arithmetic Logic Units (ALUs) in the Graphics Data Controller

Each ALU logically combines the byte written by the CPU and the byte stored in the matching latch, according to the setting of bits 3 and 4 of the Data Rotate register (and the Bit Mask register as well, which I'll cover next time), and then writes the result to display memory. It is most important to understand that neither ALU operand comes directly from display memory. The temptation is to think of the ALUs as combining CPU data and the contents of the display memory address being written to, but they actually combine CPU data and the contents of the last display memory location read, which need not be the same location being modified. The most common application of the ALUs is indeed to modify a given display memory location, but doing so requires a read from that location to load the latches before the write that modifies it. Omission of the read results in a write operation that logically combines CPU data with whatever data happens to be in the latch from the last read, which is normally undesirable.

Occasionally, however, the independence of the latches from the display memory location being written to can be used to great advantage. The latches can be used to perform 4-byte-at-a-time (one byte from each plane) block copying; in this application, the latches are loaded with a read from the source area and written unmodified to the destination area. The latches can be written unmodified in one of two ways: by selecting write mode 1 (for an example of this, see the "On Graphics" article in the last issue of *PJ*) or via the Bit Mask register.

The latches can also be used to draw a fairly complex area fill pattern, with a different bit pattern used to fill each plane. The mechanism for this is: Generate the desired pattern across all planes at any display memory address. Generating the pattern requires a separate write operation for each plane so that each plane's byte will be unique. Read that memory address to store the pattern in the latches. The contents of the latches can now be written to memory any number of times by using either write mode 1 or the bit mask since they will not change until a read is performed. If the fill pattern does not require a different bit pattern for each plane, filling can be performed more easily by using the Map Mask register (and, if display memory isn't zeroed, the Set/Reset registers, of which, more next time).

The sample program in Listing 1 fills the screen with horizontal bars, then illustrates the operation of each of the four ALU logical functions by writing a vertical 80-pixel-wide box filled with solid, empty, and vertical and horizontal bar patterns over that background using each of the functions in turn. When observing the output of the sample program, remember that all four vertical boxes are being drawn with exactly the same code—only the logical function that is in effect differs from box to box.

You will observe that all graphics in the sample program are done in black-and-white by writing to all planes since this shows the operation of the ALUs most clearly. Selective enabling of planes would produce color effects; in this case, the operation of the logical functions must be evaluated on a plane-by-plane basis because only the planes enabled by the Map Mask register will be affected.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Logical function 0, data unmodified, is the standard mode of operation of the ALUs. In this mode, the CPU data is combined with the latched data by ignoring the latched data entirely. Expressed as a logical function, this could be considered CPU data ANDed with 1 (or ORed with 0). This is the mode to use whenever you want to place CPU data into display memory, replacing the previous contents entirely. It may occur to you that there is no need to latch display memory at all when the data unmodified function is selected. In the sample program, that is true, but if the bit mask is being used, the latches must be loaded even for the data unmodified function, as I'll discuss in the next article.

Logical functions 1 through 3 cause the CPU data to be ANDed, ORed, and XORed with latch data, respectively. Of these, XOR is the most useful because exclusive-ORing is a traditional way to perform animation. The uses of the AND and OR logical functions are less obvious. AND can be used to mask a blank area into display memory or to mask off those portions of a form that don't overlap an existing display memory image. OR could conceivably be used to force an image into display memory over an existing image. To be honest, though, I have yet to come up with a truly valuable application for the OR logical function; if you've got one, write and let me know what it is.

### Notes on the ALU/LATCH Sample Program

EGA settings such as the logical function select should be restored to their default conditions before the BIOS is called to output text or draw pixels. The EGA BIOS does not guarantee that it will set most EGA registers except on mode sets, and there are so many compatible BIOSes around that the code of the IBM BIOS is not a reliable guide. For instance, when the BIOS is called to draw text, it's likely that the result will be illegible if the Bit Mask register is not in its default state. Similarly, a mode set should generally be performed before exiting a program that tinkers with EGA settings.

Along the same lines, the sample program does not explicitly set the Map Mask register to ensure that all planes are enabled for writing. The mode set for mode 10 hex leaves all planes enabled, so I did not bother to program the Map Mask register, or any other register besides the Data Rotate register, for that matter. However, the profusion of compatible BIOSes means there is some small risk in relying on the BIOS to leave registers set properly. There are only a few clone chip sets, and they all seem to be compatible, so for the highly safety-conscious, the best course would seem to be to program data control registers such as the Map Mask and Read Mask explicitly before relying on their contents.

(On the other hand, any function the BIOS provides explicitly—as part of the interface specification—such as setting the palette RAM, should be used in preference to programming the hardware directly.)

The code that draws each vertical box in the sample program reads from display memory immediately before writing to display memory. The read operation loads the EGA latches. The value read is irrelevant as far as the sample program is concerned; it is necessary to perform a read to load the latches, and there is no way to read without placing a value in a register. This is a bit of a nuisance since it means that the value of some 8-bit register must be destroyed. Under certain circumstances, a single logical instruction such as XOR can be used to perform both the read to load the latches and the write to modify display memory without affecting any register, but since this requires the bit mask, it'll have to wait until next time.

All text in the sample program is drawn by EGA BIOS function 13 hex, write string. This function is also present in the AT's BIOS, but not in the XT's or PC's, and as a result is rarely used; the function is always available if an EGA is installed, however. Text drawn with this function is relatively slow. If speed is important, a program can draw text directly into display memory much faster in any given display mode. The great virtue of the BIOS write string function in the case of the EGA is that it provides an uncomplicated way to get text on the screen reliably in any mode and color, over any background.

The expression used to load DX in the TEXT_UP macro in the sample program may seem strange, but it's a convenient way to save a byte of program code and a few cycles of execution time. DX is being loaded with a word value that's composed of two independent immediate byte values. The obvious way to implement this would be with

```
MOV DL,VALUE1
MOV DH,VALUE2
```

which requires four bytes and eight cycles on the 8088. If you shift the value destined for the high byte into the high byte with MASM's shift-left operator SHL (*100h would also work) and then logically combine the values with MASM's OR operator (or the ADD operator), both halves of DX can be loaded with a single instruction, as in

```
MOV DX,(VALUE2 SHL 8) OR VALUE1
```

which takes only three bytes and four cycles. As shown, a macro is an ideal place to use this technique; the macro invocation can refer to two separate byte values, making matters easier for the programmer, while the macro itself can combine the values into a single word-sized constant.

A minor speed tip illustrated in the listing is the use of INC AX and DEC AX in the DrawVerticalBox subroutine when only AL actually needs to be modified. Word-sized register increment and decrement instructions are only one byte long and execute in 2 cycles on an 8088, while byte-sized register increment and decrement instructions are two bytes long and execute in 3 cycles. Consequently, when speed counts it is worth using a whole 16-bit register instead of the low 8 bits of that register for INC and DEC—if you don't need the upper 8 bits of the register for any other purpose.

Another speed tip is the use of registers to temporarily store all constants in the background-drawing loop. Moving a constant to a 16-bit register takes a 3-byte instruction and adding a constant to a 16-bit register takes a 4-byte instruction, while moving and adding 16-bit registers to 16-bit registers require only 2-byte instructions. The register-register operations are also at least 1 cycle faster; they are probably a good deal faster than that relative to the constant operations because of the prefetch queue, a topic I will return to later.

The latches and ALUs are central to high-performance EGA code because they allow programs to process across all four memory planes without a series of OUTs and read/write operations. It is not always easy to arrange a program to exploit this power, however, since the ALUs are far more limited than a CPU. In many instances, however, additional hardware in the EGA, including the bit mask, the set/reset features, and the barrel shifter, can assist the ALUs in controlling data. I will begin to examine these features next time.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

**Puzzles**

I've got a couple of unsolved (in one case, partially solved) puzzles this month.

First, another MASM problem. When I use macros with several arguments, the text often runs past column 80. The code still assembles properly, but printouts are a mess. Does anyone know how to continue a macro invocation (or a STRUC initialization, for that matter) on the next line?

The second puzzle pops up regularly in coding high-performance graphics drivers—how the heck do you determine just what's the fastest 8088 code for a given purpose? Hand-optimization of Assembler code is always an art, but the prefetch queue makes 8088 optimization black magic. (See my article "More Optimizing For Speed," *PJ*, July/August 1986, for an introduction to prefetch queue effects.) Optimizing 80286 code is somewhat less critical, both because the 8088 is far more common right now and because ATs are faster than PCs no matter how fast 8088-oriented code is, but 80286 optimization is becoming more important, so that's an issue too. And, of course, 80386 optimization looms on the horizon.

I ran across a particular optimization puzzle while implementing Bresenham's line-drawing algorithm: how best to rotate a pixel mask (one bit set and the others reset) to the right (moving one pixel to the right), incrementing the screen address by one when the set bit in the pixel mask wraps around from bit 0 to bit 7? The standard code for this instance would be

```
        ROR AL,1
        JNC NO_INC
        INC DI
NO_INC:
```

where AL is the pixel mask (with only one bit set to 1), and DI is the screen address of the byte containing the next pixel to be drawn. However, JNC is executed 7 of 8 times the above code is executed, requiring 16 cycles each time, for an average code execution time for the conditional increment portion of this code of ( (16 * 7) + (4 + 2) ) / 8 = 14.75 cycles on the 8088. (The conditional increment code is JNC NO_INC / INC DI; the ROR AL,1 instruction is not counted because I'm interested only in the relative conditional increment timings of the approaches I'll examine.) On the other hand, the less elegant looking code:

```
        ROR AL,1
        JC DO_INC
CONTINUE_AFTER_INC:
        :
        :
DO_INC:
        INC DI
        JMP CONTINUE_AFTER_INC
```

where DO_INC is outside the loop the rotate is performed in but within range of a short jump, requires only ( (7 * 4) + (16 + 2 + 15) ) / 8 = 7.6 cycles to perform the conditional increment on average. (Thanks to Dan Illowsky for this unorthodox but effective approach.) The speed of this code is a result of the vast difference on the 8088 between a conditional jump taken and one not taken.

The fastest code, however, is that which eliminates branching entirely. To wit:

```
        ROR AL,1
        ADC DI,0
```

Continued...


TURBO SOURCE SEARCH (TSS)

**TSS** is a **RBBS-by-mail** for **TURBO** Pascal. Stop paying AT&T, MCI, SPRINT et al. 10 times what it will cost you if you get it by mail from TSS.

Join **TSS** for $25. Then buy our library disks, **$2 per disk.** (45 disks and growing!!) Archived files = **700+ Kb** of code per **TSS** disk. **TSS** gives more bytes/buck! **Why pay more to get less?**

For your $25 you get
- 2 **FREE** library disks (utilities & catalog)
- 2 **FREE TSS** disks with your first order
- a validated password for the **online TSS-BBS**

**TSS-BBS** will be an "e-mail" drop-off for members to place orders then log off. It will also serve more traditional RBBS needs. Members will be able to DL the **TSS** disks **(at 10 times the cost of mails!)** if they really MUST have it now!

Join the **TSS BBS-thru-the-mail** for the best in public domain and shareware **TURBO** code.

TURBO SOURCE SEARCH
P.O. Box 876, Scituate, MA 02066, (617) 545-6677
payment: VISA/MC/COD if validated, otherwise check/money orders

Number 35 on the Reader Service Card

$99 BUSINESS BOARD

24 hour business information center using modems at 300/1200/2400 baud. Has a multi-level menu system that is easily customized with no programming req'd. Includes remote PC operations and integrated data base management. Source code additional charge.

$99 PC COMMX $119 CP/M

Emulates VT100, Wyse, HP, ADM, TV, IBM, ADDS, file transfers: KERMIT, XMODEM, COMMX mainframe, Telex/TWX. Instant DOS/foreground switch! Unattended macro controls and 700 entry dial directory. Electronic mail sub-system!

$59 C DATA ENCRYPTION

Data Encryption Standard (U.S. government standard FIPS PUB46) in Microsoft "C". Includes compression & telecomm formatting, allowing for faster transmission & storage on any computer or service. Complete "C" source code provided for additional $249.

HAWKEYE GRAFIX Inc
Box 1400, Oldsmar Florida 33557
Call 813-786-8161

Number 57 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

requires exactly 4 cycles to perform each conditional increment—more than 3 times faster than the original code. This is an excellent example of the way in which the quirks of the 8088 make it possible to hand-optimize small portions of Assembler code to a far greater extent than is true of most processors.

So what's the puzzle? The puzzle is that I still don't know what's really the fastest way to perform the above rotate and conditional increment. The performance estimates above were arrived at by adding the execution times Intel publishes, but the published times assume that the prefetch queue isn't empty. For example, since ROR AL, 1 / ADC DI,0 constitutes 6 bytes, since the 8088 prefetch queue is only 4 bytes long, and since 4 cycles are required to fetch each instruction byte, there's no way that particular code fragment could ever execute in only the specified 6 cycles, even if the prefetch queue were full when ROR began executing. If the prefetch queue is empty, the ADC DI,0 instruction alone will require 16 cycles (4 cycles times 4 bytes), a far cry from the 4 cycles assumed above, and, in fact, exactly as long as the nominal time required to make a conditional jump. The time required to jump is unclear as well because the prefetch queue is emptied by each jump and may take additional time beyond the 16 cycles specified to refill before the instruction branched to can start execution. All the code fragments above are likely to take somewhat longer than specified; just how much longer is the question.

So. Here's what I'd like to know (and share with *PJ*'s readers). First, what is the precise overhead of branching and flushing the queue? Are additional cycles beyond the published timings required to get enough bytes to begin executing the next instruction?

Second, how long does a conditional jump not taken take to execute; is the second byte of the instruction even read? If so, is it read while the jump condition is evaluated?

Third, how important, in general, is the prefetch queue in determining performance of 80286/80386 code? The 80286 can fetch twice as many instruction bytes per memory access as the 8088, but it also executes most instructions in many fewer clock cycles and so uses up instruction bytes at a faster rate.

Fourth (and most important), what techniques have *PJ* readers come up with for evaluating the performance of time-critical code? Ideally, Intel would make available all the information about the 8088's microcode required to accurately determine execution times, but I know of no such publication. There are a number of code profilers out there, but most of them measure gross effects by reporting overall time spent in a given routine. To truly optimize, it's necessary to understand where every cycle goes, and to do that, each instruction in the innermost time-critical loops has to be examined in the context of normal operation. There's hardware out there that can do this; sadly, such equipment is outside the price range of many developers, especially individuals. On behalf of the under-funded developers of the world, I'd love to hear about a way to do cycle-by-cycle in-context performance analysis in software.

Who cares about a few cycles here and there? In a time-critical code path, any programmer who cares about turning out a superior product should, and on the PC, virtually all real-time graphics are time-critical. Let us know how you do it. Make the world of 8088 programming a little better.

### Next Time

I'm going to explore a couple of interesting EGA topics in the next article. I'll continue examining the data flow around the ALUs, including the bit mask, data rotate, and set/reset capabilities, which are crucial to pixel-aligned graphics and fast color control. I'll also talk about color versus monochrome operation and downward compatibility with the CGA, MDA, and Hercules Graphics Card, plus the usual discussion of interesting hardware and/or software I've come across. See you then. □

*Michael Abrash is a Senior Software Engineer for Orion Industries of Redwood City, CA, a manufacturer of PC-based instrumentation and microprocessor development systems.*

*Code follows.*


# Here's why you should choose Periscope as your debugger...

**You'll get your programs running fast.** "It works great! A problem we had for three weeks was solved in three hours," writes Wade Clark of MPPi, Ltd.

**You'll make your programs solid.** David Nanian says, "I can't live without it!! BRIEF, a text editor my company wrote, would not be as stable as it is today without Periscope."

**You'll protect your investment.** We won't forget you after the sale. You'll get regular software updates, including a FREE first update and notice of later updates. You'll get technical help from Periscope's author. And you'll be able to upgrade to more powerful models of Periscope if you need to. One Periscope user writes, ". . . your support has won over even the heart of this hardened programmer!"

**You deserve the best.** Thousands of programmers rely on the only debugger that PC Tech Journal has ever selected as **Product of the Month** (1/86). You owe it to yourself to find out why, first hand.

**You can try it at no risk.** You get an unconditional 30-Day, Money-Back Guarantee, so you can't lose.

**Start saving time and money now — order toll-free, 800/722-7006.** Use MasterCard, Visa, COD, or a qualified company purchase order. As one user puts it, Periscope is "one of the rare products, worth every penny!"

**New Version 3 is better than ever!**

| | |
|---|---|
| Periscope I, software, manual, protected memory board and breakout switch | $345 |
| Periscope II, software, manual, and breakout switch | $175 |
| Periscope II-X, software and manual | $145 |

Add shipping - $3 US; $8 Canada; $24 elsewhere.

The PERISCOPE Company, Inc.

14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

Number 6 on the Reader Service Card




PRODUCED 2006 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Code from Michael Abrash's "Parallel Processing ..."

1

```
; Program to illustrate operation of ALUs and latches in the EGA's
;  Graphics Data Controller.  Draws a variety of patterns against
;  a horizontally striped background, using each of the 4 available
;  logical functions (data unmodified, AND, OR, XOR) in turn to combine
;  the images with the background.
; Assembled with MASM 4.0, linked with LINK 3.51.
; By Michael Abrash, 11/27/86.
;
stack   segment para stack 'STACK'
        db      512 dup(?)
stack   ends
;
EGA_VIDEO_SEGMENT       equ     0a000h  ;EGA display memory segment
SCREEN_HEIGHT           equ     350
SCREEN_WIDTH_IN_BYTES   equ     80
DEMO_AREA_HEIGHT        equ     336     ;# of scan lines in area
                                        ; logical function operation
                                        ; is demonstrated in
DEMO_AREA_WIDTH_IN_BYTES equ    40      ;width in bytes of area
                                        ; logical function operation
                                        ; is demonstrated in
VERTICAL_BOX_WIDTH_IN_BYTES equ 10      ;width in bytes of the box used to
                                        ; demonstrate each logical function
;
; EGA register equates.
;
GDC_INDEX       equ     3ceh    ;GDC index register
GDC_ROTATE      equ     3       ;GDC data rotate/logical function
                                ; register index
GDC_MODE        equ     5       ;GDC mode register index
;
dseg    segment para common 'DATA'
;
; String used to label logical functions.
;
LabelString     label   byte
        db      'UNMODIFIED    AND       OR        XOR   '
LABEL_STRING_LENGTH     equ     $-LabelString
;
; Strings used to label fill patterns.
;
FillPatternFF   db      'Fill Pattern: 0FFh'
FILL_PATTERN_FF_LENGTH  equ     $ - FillPatternFF
FillPattern00   db      'Fill Pattern: 000h'
FILL_PATTERN_00_LENGTH  equ     $ - FillPattern00
FillPatternVert db      'Fill Pattern: Vertical Bar'
FILL_PATTERN_VERT_LENGTH        equ     $ - FillPatternVert
FillPatternHorz db      'Fill Pattern: Horizontal Bar'
FILL_PATTERN_HORZ_LENGTH equ    $ - FillPatternHorz
;
dseg    ends
;
; Macro to set indexed register INDEX of GDC chip to SETTING.
;
SETGDC  macro   INDEX, SETTING
        mov     dx,GDC_INDEX
        mov     ax,(SETTING SHL 8) OR INDEX
        out     dx,ax
        endm
;
;
; Macro to call BIOS write string function to display text string
;  TEXT_STRING, of length TEXT_LENGTH, at location ROW,COLUMN.
;
TEXT_UP macro   TEXT_STRING, TEXT_LENGTH, ROW, COLUMN
        mov     ah,13h                  ;BIOS write string function
        mov     bp,offset TEXT_STRING   ;ES:BP points to string
        mov     cx,TEXT_LENGTH
        mov     dx,(ROW SHL 8) OR COLUMN        ;position
        sub     al,al                   ;string is chars only, cursor not moved
        mov     bl,7                    ;text attribute is white (light gray)
        int     10h
```

2

```
        endm
;
cseg    segment para public 'CODE'
        assume  cs:cseg, ds:dseg
start   proc    near
        mov     ax,dseg
        mov     ds,ax
;
; Select 640x350 graphics mode.
;
        mov     ax,010h
        int     10h
;
; ES points to EGA memory.
;
        mov     ax,EGA_VIDEO_SEGMENT
        mov     es,ax
;
; Draw background of horizontal bars.
;
        mov     dx,SCREEN_HEIGHT/4
                                ;# of bars to draw (each 4 pixels high)
        sub     di,di           ;start at offset 0 in display memory
        mov     ax,0ffffh       ;fill pattern for light areas of bars
        mov     bx,DEMO_AREA_WIDTH_IN_BYTES / 2 ;length of each bar
        mov     si,SCREEN_WIDTH_IN_BYTES - DEMO_AREA_WIDTH_IN_BYTES
        mov     bp,(SCREEN_WIDTH_IN_BYTES * 3) - DEMO_AREA_WIDTH_IN_BYTES
BackgroundLoop:
        mov     cx,bx           ;length of bar
    rep stosw                   ;draw top half of bar
        add     di,si           ;point to start of bottom half of bar
        mov     cx,bx           ;length of bar
    rep stosw                   ;draw bottom half of bar
        add     di,bp           ;point to start of top of next bar
        dec     dx
        jnz     BackgroundLoop
;
; Draw vertical boxes filled with a variety of fill patterns
;  using each of the 4 logical functions in turn.
;
        SETGDC  GDC_ROTATE, 0           ;select data unmodified
                                        ; logical function...
        mov     di,0
        call    DrawVerticalBox         ;...and draw box
;
        SETGDC  GDC_ROTATE, 08h         ;select AND logical function...
        mov     di,10
        call    DrawVerticalBox         ;...and draw box
;
        SETGDC  GDC_ROTATE, 10h         ;select OR logical function...
        mov     di,20
        call    DrawVerticalBox         ;...and draw box
;
        SETGDC  GDC_ROTATE, 18h         ;select XOR logical function...
        mov     di,30
        call    DrawVerticalBox         ;...and draw box
;
; Reset the logical function to data unmodified, the default state.
;
        SETGDC  GDC_ROTATE, 0
;
; Label the screen.
;
        push    ds
        pop     es      ;strings we'll display are passed to BIOS
                        ; by pointing ES:BP to them
;
; Label the logical functions, using the EGA BIOS's
;  write string function.
;
        TEXT_UP LabelString, LABEL_STRING_LENGTH, 24, 0
;
; Label the fill patterns, using the EGA BIOS's
;  write string function.
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
        TEXT_UP FillPatternFF, FILL_PATTERN_FF_LENGTH, 3, 42
        TEXT_UP FillPattern00, FILL_PATTERN_00_LENGTH, 9, 42
        TEXT_UP FillPatternVert, FILL_PATTERN_VERT_LENGTH, 15, 42
        TEXT_UP FillPatternHorz, FILL_PATTERN_HORZ_LENGTH, 21, 42
;
; Wait until a key's been hit to reset screen mode & exit.
;
WaitForKey:
        mov     ah,1
        int     16h
        jz      WaitForKey
;
; Finished.  Clear key, reset screen mode and exit.
;
Done:
        mov     ah,0    ;clear key that we just detected
        int     16h
;
        mov     ax,3    ;reset to text mode
        int     10h
;
        mov     ah,4ch  ;exit to DOS
        int     21h
;
start   endp
;
; Subroutine to draw a box 80x336 in size, using currently selected
;  logical function, with upper left corner at the display memory offset
;  in DI.  Box is filled with four patterns.  Top quarter of area is
;  filled with 0FFh (solid) pattern, next quarter is filled with 00h
;  (empty) pattern, next quarter is filled with 33h (double pixel wide
;  vertical bar) pattern, and bottom quarter is filled with double pixel
;  high horizontal bar pattern.
;
; Macro to draw a column of the specified width in bytes, one- quarter
;  of the height of the box, with the specified fill pattern.
;
DRAW_BOX_QUARTER        macro   FILL, WIDTH
        local   RowLoop, ColumnLoop
        mov     al,FILL                 ;fill pattern
        mov     dx,DEMO_AREA_HEIGHT / 4 ;1/4 of the full box height
RowLoop:
        mov     cx,WIDTH
ColumnLoop:
        mov     ah,es:[di]      ;load display memory contents into
                                ; GDC latches (we don't actually care
                                ; about value read into AH)
        stosb                   ;write pattern, which is logically
                                ; combined with latch contents for each
                                ; plane and then written to display
                                ; memory
        loop    ColumnLoop
        add     di,SCREEN_WIDTH_IN_BYTES - WIDTH
                                ;point to start of next line down in box
        dec     dx
        jnz     RowLoop
        endm
;
DrawVerticalBox proc    near
        DRAW_BOX_QUARTER        0ffh, VERTICAL_BOX_WIDTH_IN_BYTES
                                        ;first fill pattern: solid fill
        DRAW_BOX_QUARTER        0, VERTICAL_BOX_WIDTH_IN_BYTES
                                        ;second fill pattern: empty fill
        DRAW_BOX_QUARTER        033h, VERTICAL_BOX_WIDTH_IN_BYTES
                                        ;third fill pattern: double-pixel
                                        ; wide vertical bars
        mov     dx,DEMO_AREA_HEIGHT / 4 / 4
                                ;fourth fill pattern: horizontal bars in
                                ; sets of 4 scan lines
        sub     ax,ax
        mov     si,VERTICAL_BOX_WIDTH_IN_BYTES  ;width of fill area
```

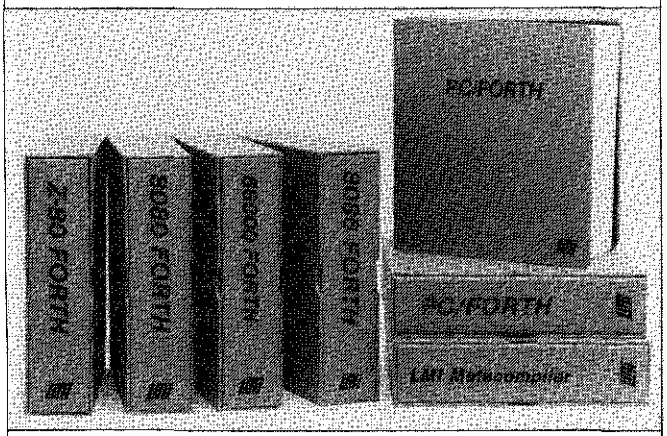


```
HorzBarLoop:
        dec     ax              ;0ffh fill (faster to do word than byte DEC)
        mov     cx,si           ;width to fill
HBLoop1:
        mov     bl,es:[di]      ;load latches (don't care about value)
        stosb                   ;write solid pattern, through ALUs
        loop    HBLoop1
        add     di,SCREEN_WIDTH_IN_BYTES - VERTICAL_BOX_WIDTH_IN_BYTES
        mov     cx,si           ;width to fill
HBLoop2:
        mov     bl,es:[di]      ;load latches
        stosb                   ;write solid pattern, through ALUs
        loop    HBLoop2
        add     di,SCREEN_WIDTH_IN_BYTES - VERTICAL_BOX_WIDTH_IN_BYTES
        inc     ax              ;0 fill (faster to do word than byte DEC)
        mov     cx,si           ;width to fill
HBLoop3:
        mov     bl,es:[di]      ;load latches
        stosb                   ;write empty pattern, through ALUs
        loop    HBLoop3
        add     di,SCREEN_WIDTH_IN_BYTES - VERTICAL_BOX_WIDTH_IN_BYTES
        mov     cx,si           ;width to fill
HBLoop4:
        mov     bl,es:[di]      ;load latches
        stosb                   ;write empty pattern, through ALUs
        loop    HBLoop4
        add     di,SCREEN_WIDTH_IN_BYTES - VERTICAL_BOX_WIDTH_IN_BYTES
        dec     dx
        jnz     HorzBarLoop
;
        ret
DrawVerticalBox endp
cseg    ends
        end     start
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# TOTAL CONTROL
## with *LMI FORTH*™




## For Programming Professionals:
## an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

### For Development:
### Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

### For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

### For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

### Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

***Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.***

**LMI** ***Laboratory Microsystems Incorporated***
*Post Office Box 10430, Marina del Rey, CA 90295*
*Phone credit card orders to:* ***(213) 306-7412***

**Overseas Distributors.**
*Germany:* Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
*UK:* System Science Ltd., London, 01-248 0962
*France:* Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
*Japan:* Southern Pacific Ltd., Yokohama, 045-314-9514
*Australia:* Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card

# TASKVIEW™

***WHY GIVE UP. . .***

- ***BATCH FILES,***
- ***I/O REDIRECTION,***
- ***SIDEKICK,***
- ***DOS MENU PROGRAMS,***
- ***MOST OF YOUR RAM,***
- ***EXECUTION SPEED?***

***Compatible, efficient DOS multi-tasking.***

*We designed Taskview with efficiency in mind. During normal operation, TASKVIEW hides behind DOS, providing you with control of up to 10 concurrent or non-concurrent programs. Just the touch of a key instantly switches a program to the foreground. Included desktop utilities let you cut and paste from program to program. Simple to use and reasonably priced, no well equipped PC user should be without it.*

*Requires: PC/AT/Jr compatible, DOS 2.0-3.1, 256K RAM, 1 Floppy drive.*

*Taskview trademark of Sunnyhill Software*
*Sidekick registered trademark of Borland Intl.*

***$69.95*** *plus $5.00 S&H*

*Washington residents add 7.9%*
*International orders add $5.00*
*VISA and Mastercard accepted.*

***30-day money back guarantee***
***Dealer Inquiries Invited.***

*To order Toll-Free*
*call 1-800-367-0651*

***Sunny Hill Software***
*13732 Midvale N. Ste. 206*
*Seattle, WA 98133*
*(206) 367-0650 M-F 8-6 PDT*

# TURBO PROFESSIONAL™

***SERVICE INTERRUPTS***
*No assembly required*

***RESIDENT PROGRAMS***
*Easy, pop-up routines*

***EXECUTIVE PROGRAMS***
*Run ANY DOS program*

***DISK SECTOR I/O***
*Lowest level access*

***FAST TEXT WINDOWS***
*Virtual windowing system*

***KEYBOARD MACROS***
*Simple, powerful*

***LOTS OF EXAMPLES***
*21+ full example programs*

***MUCH MORE. . .***
*Over 140 routines in all*

*"If you never thought Turbo Pascal was a systems programming language, you've never seen Turbo Professional."*
*Darryl Rubin*
*Computer Language*

*For programs that move with technology—Turbo Professional—a truly professional library of subroutines.*

*150 page reference manual. Full source—many example programs.*
*No royalties charged for applications.*

*Requires IBM compatible, DOS version 2.0 or greater, Turbo Pascal 2.0 or greater.*

*Turbo Professional, trademark of Sunnyhill Software*
*Turbo Pascal, registered trademark of Borland International*

***$69.95*** *plus $5.00 S&H*

*Washington residents add 7.9%*
*International orders add $5.00*
*VISA and Mastercard accepted.*

***Dealer Inquiries Invited.***

*To order Toll-Free*
*call 1-800-367-0651*

***Sunny Hill Software***
*13732 Midvale N. Ste. 206*
*Seattle, WA 98133*
*(206) 367-0650 M-F 8-6 PDT*

Number 85 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Portability in Practice and Using Header Files

by
**William Hunt**

One of the virtues claimed for C is portability—programs written in C can be compiled and executed on a wide variety of computers. Bad coding practice can make C programs unnecessarily difficult to move to a new environment; you can find plenty of rules to follow to write portable C code. But that is not sufficient to solve the portability problem. This article is about the practical problems in writing C programs for more than one environment. We'll start with some general definitions and advice.

My definition of portable C code is that it can be moved to a different hardware and operating system environment and can be compiled, linked, and executed with changes to the source code. I want to emphasize some differences between my definition and the usual ones:

1. Any change in the environment can cause portability problems—even a new compiler release or a new version of the operating system.
2. For most serious C programs, making a whole program portable is an unreasonable goal. Individual C source files are the right level for thinking about portability.
3. Portability is relative, not absolute. What is portable code depends on the change of environments you have in mind. And portability must be balanced against other desirable qualities such as quick response or a smooth user interface.

This article and those to follow will be written from the particular viewpoint described above: writing programs for serious economic activity—for sale or for important in-house activity. If such programs do not execute quickly and provide a smooth, usable interface and bulletproof operation, they will not be purchased or used. So portability must be balanced against the performance and features we need to make a program useful.

It is important to understand some of the areas in which the portability of C programs is limited. The list below is not complete but it should give you a feeling for real-world portability problems:

**Console I/O:** The C library provides simple console I/O facilities—line buffered, echoed input of ASCII characters only and scrolling TTY output. Good interactive programs need unbuffered, unechoed single character input that recognizes all the keys on the keyboard, including arrow keys (cursor control) and function keys. Neither the C library nor common operating systems provide a standardized method for single key input.

Output is even less standardized. The IBM PC and other personal computers provide direct access to the display adapter memory that supports screen refreshing. Most UNIX systems use ASCII CRT terminals as console devices; these terminals vary in capabilities and in the escape sequences needed to invoke the features. Windowing environments, such as Topview and Microsoft Windows, make console I/O even less standardized.

**File I/O:** The stdio library functions (fopen, getc, putc. . .) are widely available and included in the ANSI Standard, but they aren't useful for most serious C programs. The Unix-like unbuffered I/O functions open, read, write, are in many C libraries, but they aren't universal and they are not standardized. Although they are vital for many applications, the ANSI committee has not included them in the proposed standard.

**File Contents:** The format of text files varies from one operating system to another. On Unix systems, a single newline character (line feed) marks the end of each line. On MS DOS systems, two characters—Carriage Return and Line Feed—mark the end of a line. On other systems, text files may not contain any explicit end-of-line markers. Word processor software often stores documents in a private format that differs from the normal text file format. So utility programs that manipulate text files must often be customized for each word processor whose files must be supported.

**Add-On Libraries:** In the IBM PC environment, add-on libraries provide window management, keyed file access (B-Tree files), and graphics support. A few of these libraries are now available for Unix systems as well, but add-on libraries will continue to be available and supported mostly in the IBM PC environment.

**Error Handling:** Operating systems vary greatly in the way they handle errors. Good applications must handle errors gracefully, but that requires tailoring error handling carefully on each system. (The trend is for MS DOS C compiler libraries to simulate Unix error handling. But without adequate documentation, such library functions are worse than useless.)

The preceding suggests that we can't fully solve the portability problem. But we can limit the costs of moving a program



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

to a new environment. I've identified the steps below as a practical approach to writing portable programs:

1. Plan for Portability before you write a program. Define the range of environments in which your program should eventually operate. Put down concrete details on RAM memory, disk space, and processor and disk speed, too. It is hard enough to make programs meet the goals explicitly defined for them. Nobody succeeds in making programs meet criteria that were never identified as goals. Define goals for performance, user interface, and other features that conflict with portability.
2. Identify potential problems before you write the code. Once you define the range of environments in which your program should run, you should learn about the specific differences. For example, if you intend to move an IBM PC based program to the Unix environment eventually, you should learn a bit about Unix console I/O before you write the IBM PC version.
3. Most of your program should be quite portable. Isolate non-portable parts in separate source files and provide a portable interface to the rest of your program. For example, the low level file I/O functions read and write are not standardized. But you can define a file I/O module with function calls you establish. Then adapting to the file I/O capabilities of a particular system requires changing only one module.
4. Use C's features to control portability problems. You can define data types with the typedef statement to define data types and the #define statement for constants. Place your definitions in header files (*.H) and #include them in source files where they are needed.
5. It is important to document all the portability problems in your programs. Comments in the source files affected are important, but it is even more important to keep track of the source files that contain non-portable code.

### Using Header Files

Whenever you write C functions to be used as tools in other source files, you should construct a header file to define constants, data types, and function return values. This makes the tools easier to use and helps prevent bugs. As you build up layers of tools, it can become a burden to keep track of what header files you need in each source file. A practical solution is to have each header file include any header file it requires.

Placing #include statements in header files may lead to including the same header file twice, producing compiler error messages. But we can use the #ifndef preprocessor statement to insure that the definitions in a header file are processed only once by the compiler. For each header file, we test for the existence of a symbol unique to that header file:

```
#ifndef UNIQUE_NAME          /* once only test */
#define UNIQUE_NAME 1
...                          /* contents of this header file */
#endif                       /* end of once only test */
```

In the myansi.h file below, the MYANSI symbol is undefined the first time the file is included, so statements following the #ifndef statement are processed. One of those statements defines the MYANSI symbol, so if the myansi.h file is included again, all the definitions will be skipped.

Even if no errors are produced, reading the same include file again slows compilation. If we know a preprocessor symbol in a header file, we can avoid reading the file again by preceding an #include statement by the #ifndef statement as shown below:

```
...
#ifndef MYANSI               /* myansi.h already included ? */
#include "myansi.h"          /*   no - include now */
#endif
...
```

The header files below illustrate all these ideas.

```
/* myansi.h  - defines ANSI C features my compiler supports */
/* This version for Version 3.00 or later - Lattice or Microsoft */
#ifndef MYANSI               /* once only test */
#define MYANSI 1
#define FPRO 1               /* function prototypes */
/* void type supported */
/* memcpy in library - supply function prototype */
char *memcpy(char *,char *,unsigned) ;

#endif                       /* end of once only test */
```

Whenever we need to declare function prototypes or return values, we include "myansi.h" and test to see if the FPRO symbol is defined. For example, we would declare the malloc function with

```
#include "myansi.h"          /* defines FPRO */
#ifdef FPRO
char malloc(unsigned) ;      /* use a function prototype */
#else
char malloc() ;              /* return value only */
#endif
```

For older versions of Lattice and Microsoft compilers, the myansi.h file below leaves FPRO undefined so that function prototypes are not used. It defines the void data type, which is not recognized by these compilers and the memcpy function. □

```
/* myansi.h  - defines ANSI C features my compiler supports */
/* Version for Lattice 2.15 and older and MSC 2.03 and older */
#ifndef MYANSI               /* once only test */
#define MYANSI 1

/* leave FPRO undefined - no function prototypes */

typedef int void ;           /* void type not supported by compiler */
                             /* memcpy not in library */
#define memcpy(to,from,n)  movmem(from,to,n)

#endif                       /* end of once only test */
```

*William Hunt writes the* Lattice C Users' Group *newsletter, where this material first appeared, and is also the author of* The C Toolbox, *published by Addison-Wesley. Readers may contact Bill at the following address:*

*William James Hunt*
*Lattice C Users' Group*
*P.O. Box 271965*
*Concord, CA 94527*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# The Proposed ANSI C Language Standard

by
**Rex Jaeschke**

©*Rex Jaeschke, 1987. All rights reserved.*

I have recently installed Microsoft C V4, Lattice C V3.0 and Wizard C V3, each of which claims to be tracking the proposed ANSI Standard—in any case, they all implement some form of function prototypes, the subject of this column. These compilers will be referred to as MSC, LATC and WIZC, respectively, throughout this article. I have also been working with C-terp 2.30b (Lattice V3 variant) and Instant-C V2.2 (Dec '86). These will be referred to as CT and IC, respectively. Other DOS C compilers are known to support function prototypes. These include Whitesmiths' V3 and High-C from MetaWare.

## Function Prototypes

The primary aim of the proposed ANSI Standards Committee is to consolidate existing practice and in particular, to standardize the runtime library. Certainly, extensions have been added, the most significant of which has been function prototypes. This concept was adapted from an idea that Harris Computers have been using for quite some time.

As the name might suggest, a function prototype (called a prototype for short) is the skeletal definition of a function. I say "skeletal" because a real function definition contains the statements that make up the function whereas a prototype contains everything but that executable code. So a prototype is really a function declaration—in fact, it is simply an extension to the way we have been declaring functions since the C time began.

Specifically, this declaration contains the function's return type and the list and type of arguments for that function. For example, the prototype declarations for some of the functions declared in **ctype.h** are

```
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
```

Prior to the existence of prototypes, these functions would have been declared as follows:

```
int islower();
int isprint();
int ispunct();
int isspace();
int isupper();
```

Actually, since these functions have the default return type **int**, they might not even be declared in programs unless the address of one or more of these functions were needed. (This is because a function must be declared if its address is to be taken, even if the function is of return type **int**.)

The prototype format declaration (or simply, prototype) includes the types and names of a dummy argument list. According to the Standard, the argument names are optional—they simply indicate where an actual argument name would go in a function call or definition. (We'll discuss "new-style" function definitions later.) However, in an earlier version of the Standard, these names did not exist, so compilers that implement that earlier version will *not* accept them there. Neither MSC nor LATC accepts the dummy names, but WIZC does. (This is an example of the problem of implementing parts of a Standard that is not yet final.) In this column, we will continue to use dummy argument names. CT accepts prototypes without dummy argument names (a la Lattice, as expected); however, it does not yet check prototypes against function calls and definitions. IC does not support prototypes at all.

The dummy variables used in prototypes have prototype scope that ends at the semicolon terminating the prototype of which they are part. They are formal argument names only and have no storage space or name-space, and the same names can be used as identifiers by the user and in other prototypes currently in scope.

The prototypes above indicate that the functions return an **int** and that they each have one int argument. Note also that the declaration is terminated with a semicolon. Except for the fact that there is no function body (within {...}), a function prototype looks just like a new-style function definition (hence the name prototype).

Why have prototypes? Well, their purpose is to allow argument list type, order, and count checking across multiple source files at compile-time. If the prototype for a function is placed in a header that is included in the file where that function is defined and in all files where it is referenced, the compiler can check that all references to the same function match each other and that of the definition.

## The const Keyword in Prototypes

Other examples of prototypes (from **stdio.h**) are

```
int remove(const char *pathname);
int rename(const char *old, const char *new);
FILE *fopen(const char *pathname, const char *type);
void setbuf(FILE *stream, char *buf);
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

In some of these prototypes the new keyword **const** is used. This indicates that the corresponding variables are to remain constant; they may not be changed by the user code. This provides an opportunity for optimization by the compiler. If it knows that an argument is passed by address yet that argument has the attribute **const**, the compiler can optimize across that function call since it knows that the called function will not change the value of that argument.

Other prototypes that include const occur within the header **string.h** as follows:

```
char *strcpy(char *s1, const char *s2);
char *strcat(char *s1, const char *s2);
```

### Problems Resulting from ANSI Header Changes

Note that **string.h** is relatively new for most compilers. It contains the **str*** and **mem*** library function declarations that were previously declared in **stdio.h**. Once you convert existing code that uses these functions, you will have to include **string.h** as well so that their return values are declared properly. If you forget to do this or you don't realize you have to do this, you may or may not get an error depending on how you use these functions. For example,

```
#include <stdio.h>

...
        char *temp;
...
        temp = strcpy(dest, source);
...
```

This should work on compilers which declare **strcpy** in **stdio.h**. However, if a compiler declares **strcpy** in **string.h** and the proposed ANSI strong type-compatibility checking is implemented, you should get a syntax error. This is because **strcpy** is not declared and therefore defaults to an **int** return type, yet you are assigning that **int** to a character pointer and these types are not compatible. If you really want to do such an assignment, you must use an explicit cast. In any case, the error is really that **#include <string.h>** is missing from the file.

```
#include <stdio.h>

...
        strcpy(dest, source);
...
```

In this case, the return value from **strcpy** is not used, so it is irrelevant whether it returns an **int** or **char** *, and the absence of **string.h** is of no consequence.

The same problem can occur with numerous other library functions since they are now declared in headers other than **stdio.h**.

### Variable Argument Lists

The prototype naming conventions also allow for the specification of the leading part of an argument list. The most likely use of this is with functions that have a variable number of arguments such as the **printf** and **scanf** families declared in **stdio.h**.

```
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(char *s, const char *format, ...);
int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *s, const char *format, ...);
```

Here, all of the known (guaranteed) arguments are declared, and this list is followed by a comma and an ellipsis indicating that there may be more arguments but no information is known about their types or numbers. This format can also be used with fixed type and size argument lists, but it serves no purpose and, in fact, reduces the ability of the compiler to diagnose argument list mismatches. (See the section on Disabling Argument Widening later on for a possible use of this.)

In an earlier version of the draft Standard, the ellipsis was not used—the argument list simply ended with a comma. However, this format is no longer supported by the draft, and different compilers provide varying levels of support for this capability as follows. MSC supports the new format. It also supports the older format since that was implemented in their V3 compiler several years ago. However, if you request "maximum warnings enabled," the older format is flagged as an extension. If you are migrating from their V3 to V4 and you use variable argument prototypes, it is strongly suggested that you add the ellipsis.

LATC (and CT), on the other hand, supports only the terminating comma—it gives a syntax error if the ellipsis is present. WIZC supports only the latest format and gives a syntax error if the ellipsis is omitted.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Apart from supporting these library functions, the Standard now provides a portable way of writing and calling your own C functions from C using variable argument lists. This requires use of the header stdarg.h, which defines the macros **va_start**, **va_end**, and **va_arg** and defines the type of **va_list**. (A discussion of this capability is outside the scope of this article.) Support for variable argument passing is provided by WIZC and MSC. MSC also supplies the header **varargs.h** to maintain compatibility with XENIX (and other systems), which has supported this capability for some time. LATC (and CT) does not provide this capability.

In the next issue, I'll continue with more on function prototypes.

**Future Meetings**

The schedule for future ANSI meetings is

June 8-12, Paris, France
September 14-18, Boston, Massachusetts
December 7-11, Phoenix, Arizona □

*Rex Jaeschke is an independent computer consultant, writer and lecturer. He is also a member of the X3J11 C Standard's Committee, Co-founder and Editor of* The C Journal, *and the C language Editor of* The DEC Professional.

*If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to*

*Rex Jaeschke*
*2051 Swans Neck Way*
*Reston, VA 22091*

*Rex will address your concerns in future columns as space permits.*

**FORTRAN PROGRAMMERS**

VERSION 2.20 NOW AVAILABLE CALL FOR DETAILS

Looking for the right PC FORTRAN language system? If you're serious about your FORTRAN programming then you should be using F77L - LAHEY FORTRAN.

*"Lahey's F77L FORTRAN is the compiler of choice. It's definitely a 'Programmers FORTRAN,' with features to aid both the casual and the professional programmer. . . F77L compiled the five files in a total of 12 minutes, which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN." - PC Magazine*

Compare the features and performance of other PC FORTRANs with F77L and you will find that F77L is clearly the superior product.

- Full Fortran 77 Standard (F77L is not a subset)
- Popular Extensions for easy porting of mini and mainframe applications
- COMPLEX*16, LOGICAL*1 and INTEGER*2
- Recursion - allocates local variables on the stack
- IEEE - Standard Floating Point
- Long variable names - 31 characters
- IMPLICIT NONE
- Fast Compile - increases productivity
- Source On Line Debugger (Advanced features without recompiling)
- Arrays and Commons greater than 64K
- Clear and Precise English Diagnostics
- Compatibility with Popular 3rd Party Software (i.e. Lattice C)
- Easy to use manual
- Technical Support from LCS

- *NEW FEATURE - NAMELIST*

**F77L - THE PROGRAMMER'S FORTRAN**

**$477.00 U.S.**

System Requirements: MS-DOS or PC-DOS, 256K, math coprocessor (8087/80287)

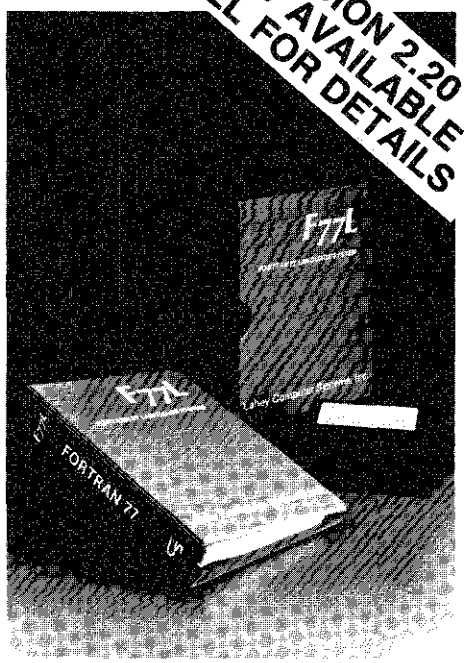
**FOR MORE INFORMATION: (702) 831-2500**

**Lahey Computer Systems, Inc.**
P.O.Box 6091
Incline Village, NV 89450
U.S.A.

**International Dealers:**
England: Grey Matter Ltd., Tel: (0364) 53499
Denmark: Ravenholm Computing, Tel: (02) 887249
Australia: Computer Transitions, Tel: (03) 537-2786
Japan: Microsoftware, Inc., Tel: (03) 813-8222

*SERVING THE FORTRAN COMMUNITY SINCE 1967*

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation.Pro FORTRAN refers to Professional FORTRAN a trademark of International Business Machines.




*Editor's Choice*
*- PC Magazine*

**Number 44 on the Reader Service Card**



PRODUCED 2005 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Complete Accounting Software with Source Code

Create Custom Vertical Accounting Modules

• General Ledger • Inventory
• Accounts Payable• Cost Accounting
• Accounts Receivable

Buy all five software modules for $200 or purchase them separately for $50 each. Each module comes complete with full documentation, an executable file and documented pretty-printed source code.The modules share an easy-to-use common user interface and work together to make accounting tasks simple. A low-cost OEM license is now available for software developers.

System requirements: Turbo® Pascal 3.0 or above, the Turbo Pascal Toolbox, and an IBM® PC or compatible. Each module is shipped on a DSDD 5 1/4" diskette. Shipping charges: $5 US, $10 Canada. Send check or money order to:

PS BUSINESS SYSTEMS

PS Business Systems
4400 Wallingford Ave N.
Suite 3
Seattle, WA 98103
(206) 632-4568

Number 82 on the Reader Service Card

# LUCID, INC.

Learning UNIX/C In-Depth

**"FOR THE CLARITY YOU EXPECT FROM TRAINING"**

**Ask about our HANDS-ON course for "C" Program Development Under MS/DOS"**

**For Free Brochure, Information or Registration Please Write or Call:**
**Alice Moss — Director**

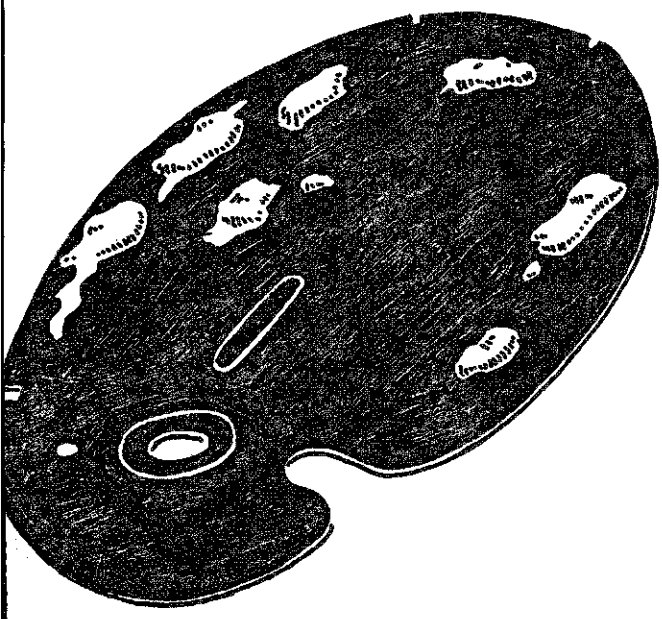
## LUCID, INC.

260 Fifth Avenue
New York, N.Y. 10001
Suite 901

(212) 685-7106

Unix is a trademark of AT & T Bell Labs
MS-DOS is a trademark of Microsoft Corp.

Number 59 on the Reader Service Card



# PC/VI

*Full Screen Editor for MS-DOS (PC-DOS)*

Looking for an Ultra-Powerful Full-Screen editor for your MS-DOS or PC-DOS system? Are you looking for an editor FULLY COMPATIBLE with the UNIX*VI editor. Are you looking for an editor which not only runs on IBM-PC's and compatibles, but ANY MS-DOS system? Are you looking for an editor which provides power and flexibility for both programming and text editing? If you are, then look no further because **PC/VI** IS HERE!

The following is only a hint of the power behind **PC/VI:** English-like syntax in command mode, mnemonic control sequences in visual mode; full undo capability; deletions, changes and cursor positioning on character, word, line, sentence, paragraph or global basis; editing of files larger than available memory; powerful pattern matching capability for searches and substitutions; location marking; joining multiple lines; auto-indentation; word abbreviations and MUCH, MUCH MORE!

The **PC/VI** editor is available for IBM-PC's and generic MS-DOS based systems for only $149. For more information call or write:

Custom Software Systems
P.O. Box 678
Natick, MA 01760
617-653-2555

The UNIX community has been using the VI editor for years. Now you can run an implementation of the same editor under MS-DOS. Don't miss out on the power of **PC/VI!**

*UNIX is a trademark of AT&T Bell Laboratories.

Number 66 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Indexing in BASIC

by
**Ethan Winer**

One of the main goals of programming has always been to create code that runs as quickly as possible while occupying the least amount of memory. I have always felt that much of the art of our profession revolves around these two goals. Indeed, there are probably as many approaches to writing efficient programs as there are programmers.

One important area where a program can really slow down is in retrieving data from a random access disk file. Unless extra steps are taken, searching for a particular name or account number requires reading each record starting at the beginning of the file. When thousands of records are involved, the time can become considerable.

The most common technique for enabling quick searching is to employ an indexing system, and in this discussion we will look at some ways to achieve this using BASIC. While all of the techniques I will be presenting can be applied to any version of BASIC for the PC, Microsoft's QuickBASIC 2 provides several distinct advantages. Most notable is its use of large numeric arrays that can occupy all of memory.

### Indexing Numbers

In any indexing scheme, the key objective is to maintain a table of information in memory that will then help locate the data in a disk file. For example, in a database of customer accounts, you would most often need to search by an account number. If a table of these numbers is also kept in an array in memory, they can be searched that much faster.

The most straightforward approach would be to have each element in the array correspond with one record in the disk file. When the database is to be searched for a particular account number, the array in memory is examined instead. If the correct number is found at, say, element 1200, the program can go directly to record 1200 on the disk and then put the complete information on the screen.

This system works well for simple numbering schemes, and even better if the numbers can be contained in the range allowed by integers (0 to 32767). Accommodating account numbers such as 4309-A is of course possible, but will require a bit more effort on your part as well as slightly more memory.

Let's begin by looking at a program example that will perform a simple integer lookup. All of the code fragments in this article will use syntax intended for QuickBASIC 2. Since you wouldn't be reading this unless you were serious about programming in BASIC, it is assumed that you already own QuickBASIC. And if you don't, perhaps these examples will induce you to buy it.

Since QuickBASIC allows the use of line labels as well as numbers (or neither for that matter), when the program's user chooses "Search" from your menu, you could use GOTO SEARCH. For clarity, we'll assume that these variables have already been defined as integers. While not essential, integer operations are infinitely faster and use half the memory of regular numeric variables. We'll also assume that the disk data has already been opened as file #1 and fielded for the appropriate variables.

```
SEARCH:

    Input "Account number to find: ", A
    Record = 0
    For X = 1 To Number.Of.Accounts
        If Array(X) = A Then
            Record = X
            X = Number.Of.Accounts
        End If
    Next
    If Record = 0 Then Print "Not Found" : Goto Err.Handler
    Get #1, Record
    Goto DISPLAY.RECORD
```

Notice the extra step where the counter X was set equal to the number of accounts being searched. It's not a good idea to leave a For/Next loop still pending, and doing it this way will still retain the found element number. Also notice that even though the account numbers are stored sequentially in memory, the actual numbers do not need to be sequential. That is, record 1—and Array(1)—could hold the account number 16032 just as easily as it could any other account number.

By the way, an easy way to determine how many records are in a file is to divide the length of the file by the length of each record. This eliminates having to retain the number separately somewhere else.

```
Open File.Name$ As #1 Len = 92
Number.Of.Accounts = LOF(1) / 92
```

Be careful with older versions of the BASCOM compiler, though. BASCOM 1 (and early versions of the BASIC interpreter, too) always wrote disk files in multiples of 128 bytes. Unless each record is also a multiple of 128, you'll end up with numbers like 233.5682, which isn't very useful.

Of course, searching is not the only aspect of maintaining an index—you'll also have to keep a copy of the index on disk and update it whenever changes are made to the main database. There are a couple of approaches possible for loading and saving this index file—the simplest would be to store the account numbers in a separate random file. At the beginning of each session, the entire file could be read into the array. If an account number were changed, the record would be updated for both files. And if a new account were added, an additional record would be written to the end of each file.

Another method is to store the index on disk as a binary image of the array. Using Bsave and Bload is considerably faster than normal read and write file operations, though at the



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

expense of extra complication. The very fact that QuickBASIC allows many, separate large arrays means that your program must determine both the offset and the segment of these arrays. Varptr will provide this information, but you must also use a special routine on the QuickBASIC disk to split the address into these two parts. It will be up to you to determine whether using Bsave and Bload is worth the extra trouble.

If you decide to use the Bsave method, you determine the length of the memory image to save by multiplying the number of elements by the size of each element. If the index is being stored in an integer array, each element will occupy two bytes. For single precision arrays, four bytes are used to store each element. In either case, the PTR86 function must be used to find exactly where in memory the array is located.

The last maintenance chore required is handling deleted records. There are many different ways to achieve this though most programmers mark the affected record in some way. If all of your searches, reports, and updates are performed using the index, only the index needs to be marked, perhaps by setting the account number to zero, or minus one. Otherwise, you might also fill the name field in the main file with asterisks or clear an otherwise mandatory field to blanks. When the range of legitimate values must include all numbers, an extra byte will be required for each record in the file, just to indicate whether or not it is deleted.

### Indexing Text Fields

So far we've limited the discussion to numeric fields that can be stored as integers. Even if a larger range of numbers is required (such as zip codes), many single precision values can still be stored internally. But what happens when a text field such as City must be indexed? While you could certainly retain quite a few index strings in memory, things would quickly become complicated when many records were involved. Unless you were willing to accept a limit on the number of records, the program would have to load the index strings into memory in sections and keep track of which portion was in memory at any given time. If a match isn't found when searching, additional sections of the index must be read in and examined until the entire file has been searched.

One possible solution to that mess is to develop a "checksum" based on the sum of the ASCII values of each character in the text field. If this checksum were stored in a large integer array, chances are one in several thousand that you would find the correct one on the first try. If, after reading the record from disk, it was found that the wrong one had been located, all your program would have to do is continue searching the array. One limitation with this approach is that wildcards could not be accommodated. Another is that to insure consistent results, the checksum must be derived after temporarily capitalizing the field. Let's elaborate.

For each City field in the database you would have a corresponding entry in a second random file that might be named CITIES.NDX. To generate each checksum in the index file would merely require adding the ASCII values for all of the characters in the field. Below is a QuickBASIC subprogram that will convert a passed string into the required checksum while using a capitalized version of the string.

```
Sub CheckSum(X$, Sum) Static

     Static CS, X, Temp             'local variables
     CS = 0
     For X = 1 To Len(X$)
         Temp = Asc(Mid$(X$, X))
         If Temp > 96 And Temp < 123 Then Temp = Temp - 32
         CS = CS + Temp
     Next
     Sum = CS

End Sub
```

Again, if all of the program's work variables are integers, the program will run that much faster.

Since QuickBASIC allows true parameter passing, any string could be converted using the same subprogram, as shown below.

```
CITY.SEARCH:

     Line Input "City to search for: ", City$
     Call CheckSum(City$, CSum)
     Record = 0
     For X = 1 To Number.Of.Accounts
         If City(X) = CSum Then
            Get #1, X
            If File.City$ = City$ Then
               Record = X
               X = Number.Of.Accounts
            End If
         End If
     Next
     If Record = 0 Then Print "Not Found" : Goto Err.Handler
```

Of course, indices are generally created only once. From then on, the index file will need to be updated only when a change in the main data file takes place. Once an index has been created this way, a search routine could then call the subprogram above, specifying the search city just prior to scanning the index array.

When really large files are involved, yet another approach may be useful. If you split up an index file into several parts and store each on disk, the amount of time spent searching can be reduced even further. For example, I designed a billing/receivables system for a local firm that has many thousands of accounts. With that many customers, it is impractical to expect a secretary to look up the account number each time a customer calls with an order or a question about billing. Instead, the name field was indexed, and a separate file was created for each letter in the alphabet, that is, A-NAMES.NDX, B-NAMES.NDX, etc.

I didn't use the checksum method (actually, I hadn't thought of it yet), but this approach has proved very satisfactory. When Jane Smith calls with a question about her account, her name is entered (Smith, Jane), and the program opens the appropriate index file (S-NAMES in this case). Each record is then searched in sequence until the correct one is found. Since the name field in this particular database is 30 characters long, each record in the index file has a length of 32. The last two bytes contain the record number in the main file, and the number is stored as an integer in MKI$ form.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Notice that since multiple index files are used, there can no longer be a one-to-one relationship between the record numbers in the main and index files. Therefore, two extra bytes are also required in the main file to hold the corresponding record number in the appropriate index file.

Again, the entries for this index file should always be written after capitalizing the field, and the name to find also capitalized just prior to searching.

### Indexing Date Fields

There are a few other tricks you can use to help speed up searching, depending on the type of field being indexed. For example, if information is often retrieved based on a date or range of dates, you will probably want to reverse the order of the digits, as well as pack the dates to the smallest possible size. After all, the smaller the file, the faster it can be searched.

Normally, dates are stored as MMDDYY, which precludes a direct comparison. That is, 123186 will appear to be later than 010187 regardless of whether you use a numeric or string comparison. At the very least, before a date is stored in an index file the digits should be swapped around into YYMMDD form. Better still, pack each pair of digits into a corresponding CHR$ value. This way, each date will occupy only three bytes, as opposed to four for a single precision number, or six for a string.

The two functions below will convert dates between the string form MMDDYY and a packed ASCII format of YMD.

```
Def FnPack$(Dt$) = Chr$(Val(Right$(Dt$, 2   ))) + _
                   Chr$(Val( Left$(Dt$, 2   ))) + _
                   Chr$(Val(  Mid$(Dt$, 3, 2)))


Def FnUnPack$(Dt$)

    Static X, Temp, Temp$
    Temp$ = ""
    For X = 1 To 3
        Temp = Asc(Mid$(Dt$, X))
        If Temp < 10 Then Temp$ = Temp$ + "0"
        Temp$ = Temp$ + Mid$(Str$(Temp), 2)
    Next
    FnUnPack$ = Mid$(Temp$, 3, 2) + Right$(Temp$, 2) + _
                Left$(Temp$, 2)

End Def
```

Then, to pack the date variable DT$, just invoke the first function this way:

```
DT$ = FnPack$(DT$)
```

You can unpack it with

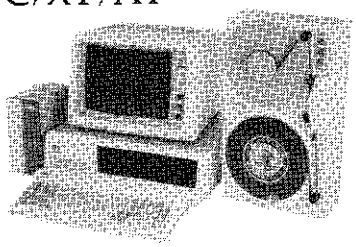
```
DT$ = FnUnPack$(DT$)
```

By the way, never try this on a sequential file. If the day happens to be the 26th of the month, you will be putting an end of file marker in the middle of the data, and it will be very difficult to get it back again later. (Believe me, I've been there!)

### Multiple Choices

Any means of compacting data will inevitably improve the


9-Track Tape Subsystem
for the IBM PC/XT/AT

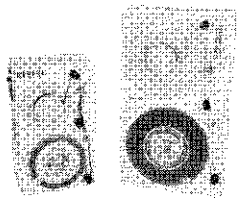


The solution to your micro/mainframe communications problem is available today!

Qualstar's new ½ inch 9-track MINISTREAMER™ brings full ANSI data interchange capability to the PC. Now you can exchange data files with virtually any other computer using 9-track tape.

Available in both 7" and 10 ½" versions, the MINISTREAMER uses less desk space than an ordinary sheet of paper, yet provides full 1600/3200 BPI capability at an affordable price. Up to 134 megabytes of data (depending on format) can be stored on a standard 10 ½" reel of tape, thus making the MINISTREAMER a highly-reliable answer to your backup requirements as well.

Tape subsystem includes tape drive, full-slot coupler card, cables and MS-DOS or XENIX compatible software.



Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9015 Eton Avenue,
Canoga Park, CA 91304
Telephone: (818) 882-5822

Number 97 on the Reader Service Card

ISIS
CP/M MSDOS

COMPLETE SOURCE CODE INCLUDED!

**ICXPDS:** eXchanger now supports the 5 1/4"iPDS format. Manipulation of ISIS-II files using your computer system was never easier.

**ICXMDS:** Same as ICXPDS, but for MDS 8" systems.

**IMXPDS:** Reads/Writes 5" iPDS disks on PC's and AT's.

**TELEDPLUS:** Enhanced serial file transfer program for CP/M, ISIS, or MS-DOS.

**ISE:** Emulator gives the CP/M and MS-DOS user access to all the ISIS-II languages and utilities.

**ACCELER 8/16:** CP/M-80 emulator for MS-DOS. Enables PC's to run ISE. (no source code, V-20 incl.) $89 each
$250 any 3 above

**UDI:** The 8086 ISIS Emulator runs all UDI applications. ..... $300

**ZAS Development Package:** Z-8 and Z-8000 Assembler for CP/M, ISIS, and MS-DOS.

*Request a catalog of our products!*

Copyrights: CP/M Digital Research, Inc. ISIS-II and iPDS Intel Corp. MSDOS Microsoft

Western Wares 303-327-4898
Box C • Norwood, CO 81423

Number 100 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

speed of a search, so the last technique I'll show here will provide the greatest saving in disk space. In all fairness I must mention that this trick is taken directly from DataEase, one of the best general purpose database programs available.

Frequently, the contents of a field can be reduced to one of several choices. Suppose you're creating a database for a manufacturer whose products fall into twelve categories. You could always use a text field long enough to hold the longest choice, but a much better way is to code each with a single number. Then, as with the dates, you could pack the choices down to an equivalent CHR$( ) byte and store that on the disk instead. Not only does this save disk space, but it also minimizes both the amount of typing and the chance for errors.

```
Dim Type$(12)
For X = 1 To 12
    Read Type$(X)
Next
Data Automotive, Farm, Plumbing, Electrical, etc.
```

Now, whenever a record is retrieved from disk, instead of saying,

```
Print Type$
```

to display it on the screen, you would instead use

```
Print Type$(Type)
```

where Type is the single byte retrieved from the disk file.

## Think Packed

We have looked at a variety of methods for indexing disk data as well as compacting that data to further speed searching. In parting, let me encourage you to be creative in this regard, and to always "think packed." Every byte saved will make a difference, as in the example account number at the beginning of this article. When you come across a numbering system that uses a format such as 4309-A, don't store it as a string and use six bytes. At the very least, strip the hyphen and show it only on the screen. And to compress it to only four bytes, convert the trailing letter to its ASCII counterpart and turn it into a single precision number: 430965.

Finally, use functions and subprograms whenever possible. If you need to do this type of conversion more than once in a program—especially on different variables—you can realize a considerable savings in program space. For example, in the City search shown earlier, you would really want to compare the inputted city with the city on disk after both had been temporarily capitalized. By first defining a function like the one shown below, you could do this comparison as often as needed with the minimum amount of code.

```
Def FnUp$(X$)
    Static X, Temp
```

Continued . . .

386•386•386•

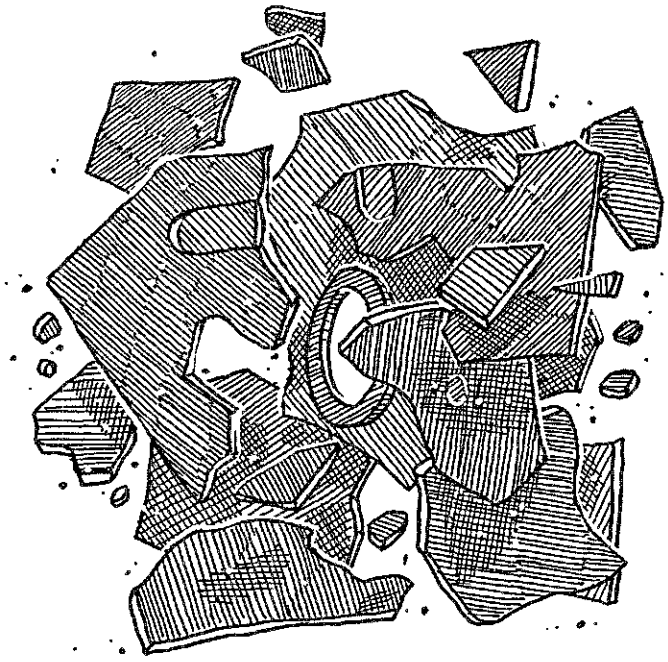
```
    For X = 1 To Len(X$)
        Temp = Asc(Mid$(X$, X))
        If Temp > 96 And Temp < 123 Then
            Mid$(X$, X, 1) = Chr$(Temp - 32)
    Next
    FnUp$ = X$

End Def
```

Then whenever a case-independent comparison had to be performed, you could use

```
If FnUp$(City$) = FnUp$(File.City$) Then . . .
```

Other user defined functions can be tremendous space savers when properly implemented, but we'll leave that discussion for next time. □

*Ethan Winer owns Crescent Software and is the author of the QuickPak utilities for QuickBASIC.*




FREE SOURCE CODE!

VITAMIN C
*It's good for your system!*

FREE SOURCE CODE!

**Vitamin C Difference**

With **Vitamin C,** your applications come alive with windows that explode into view! Data entry windows and menus become a snap. Vitamin C's **open ended design** is full of "hooks" so you can "plug in" special handlers to customize most routines. Of course, Vitamin C **includes all source code FREE,** with no hidden charges. *It always has.*

**Windows**

Create windows with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, scroll bars, sizes to 32k, and more. Unique built-in feature lets users move and resize windows at run-time!

**Data Entry**

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields, picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited validation via standard and user definable routines.

**High Level Functions**

**Standard help handler** provides context sensitive pop-up help messages any time the program awaits key strokes. So easy to use that a single function initializes and services requests by opening a window, locating, formatting, displaying and paging through the message.

**Multi-level MacIntosh & Lotus style menus** make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens quickly and easily.

**Text editor windows** can be opened for pop-up note pads and general purpose editing. Features include insert, delete, word wrap, justify, cut, paste, search, and more!

**VCScreen**

With VCScreen and Vitamin C working together, you'll reach a new level or productivity you can't reach with a function library alone!

VCScreen speeds development even more! The interactive screen editor actually lets you draw input, output and constant fields, headings, boxes, lines, even a window for your froms to run in.

VCScreen generates readable C source code ready to "plug in" to your application and link with Vitamin C.

**Guarantee**

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C . . . . . $225.00

*Includes ready to use libraries, tutorial, reference manual, demo, sample and example programs, and quick reference card; for IBM PC and compatibles. Specify compiler and version when ordering.*

Vitamin C Source . . . . . FREE*

**Free with purchase of Vitamin C.*

VCScreen . . . . . $99.95

*Requires Vitamin C and IBM PC/XT/AT or true compatible.*

Shipping $3 ground, $6 2-day air, $20 overnight, $30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on U.S. bank. Texas residents add 7¼% sales tax.

(214) 245-6090

creative PROGRAMMING
Creative Programming Consultants, Inc.
Box 112097 Carrollton, Texas 75011



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# 80386

## SOFTWARE DEVELOPMENT TOOLS

***The Phar Lap 80386 Software Development Series:***

**386|ASM/LINK** by Phar Lap (MS-DOS®) $495
Full-featured macro assembler and linker. Includes a debugger and 32-bit protected mode runtime environment.

**80386 High-C™** by MetaWare (MS-DOS®) $895

**80386 Professional Pascal™** by MetaWare (MS-DOS®) $895

**UNIX™ and VAX/VMS®** cross tools (call)

The wait for professional software development tools for the 80386 is over! Whether you are upgrading an existing IBM PC application to the '386, moving a mainframe application down to a PC, or writing the next PC best-seller, the Phar Lap 80386 Software Development Series is for you. It is an integrated line of products which provides everything you'll need to create and *run* 80386 protected mode applications under MS-DOS.

**(617) 661-1510**

Phar Lap Software, Inc. "The 80386 Software Experts"

60 Aberdeen Ave. Cambridge, MA 02138

Number 96 on the Reader Service Card

---

## ★ NOW AVAILABLE FOR C ★ QUICKBASIC ★ ★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE-**
  **ASE** is still only..... **$99**
  **DEMO** (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

**TRANSPORTABLE ★ FULLY DOCUMENTED**
**EFFICIENT ★ AFFORDABLE**

**ASP** includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

**ASP** is still only..... **$130**

**ASE** and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

ASPEN SYSTEMS

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number 86 on the Reader Service Card

---

# C CODE FOR THE PC

*source code, of course*

| | |
|---|---|
| Panache C Program Generator | $125 |
| The Profiler | $100 |
| QC88 C Compiler | $90 |
| Concurrent C | $45 |
| Biggerstaff's System Tools | $40 |
| Coder's Prolog in C | $45 |
| Translate Rules to C | $30 |
| ICON String Processing Language | $25 |
| LEX | $25 |
| YACC & PREP | $25 |
| tiny-c interpreter & shell | $20 |
| C Tools | $15 |

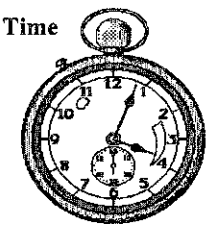
The Austin Code Works
*11100 Leafwood Lane*
*Austin, Texas 78750-3409*
*(512) 258-0785*

Free shipping on prepaid orders MasterCard/VISA

Number 30 on the Reader Service Card

---

## Along With Your Computer, Your Time is the Most Important Thing You Own. . .So Why Waste It?



***Quilt Programmer Productivity Tools will help you manage your software projects and get control of your time!***

**SRMS™**
**Software Revision Management System**

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without
- Windowing Shell interface to simplify use
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+

**SRMS Version 3.0.....$185**

**QMAKE™**
**Intelligent Program Generation Utility**

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatability, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+

**QMAKE Version 1.2.....$99**

**SRMS + QMAKE .....$250**

**NEW ! TXT Tools NEW !**

QSE - Quilt Text Stream Editor
QSRCH - Quilt File Search Utility (Like UNIX GREP)
QDIFF - Quilt Windowing File Difference Utility

**TXTTOOLS Version 1.0..... $85**

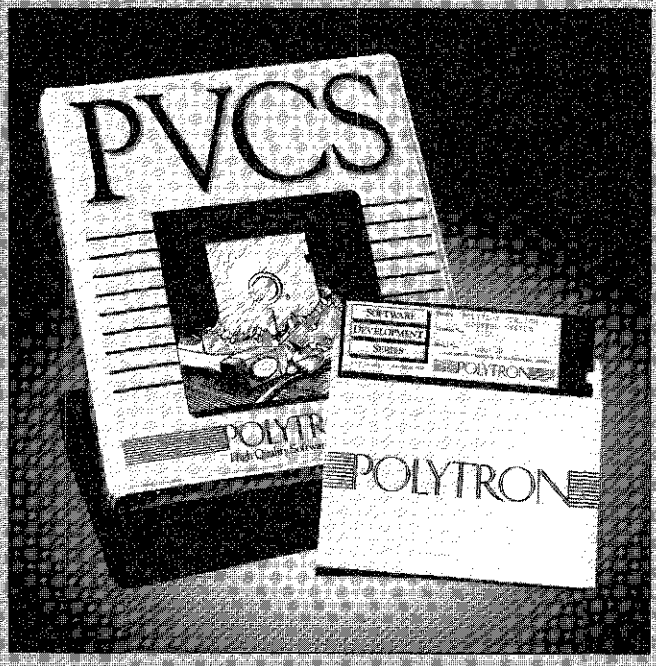
QUILT COMPUTING

7048 Stratford Road
Woodbury, MN 55125
(612) 739-4650

Volume Discounts and Dealer Inquiries Welcome

Number 46 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




# PVCS

## *The Most Powerful and Flexible Source Code Revision & Version Control System for PCs and Local Area Networks*

*Programmers, Programming Teams, Project Managers, Librarians and System Administrators Find PVCS Indispensable*

When large and complex software programs and systems are being developed on personal computers, effective management of the revisions and versions becomes critical. The POLYTRON Version Control System (PVCS) simplifies this process and lets you effectively control the proliferation of code changes. We used UNIX SCCS and RCS as models. However, our own experience, and the input of hundreds of PC programmers has enabled us to significantly improve upon these models.

**PVCS** provides many powerful functions including:

* Storage & Retrieval of multiple revisions of text.
* Maintenance of a complete history of changes.
* Maintenance of separate lines of development using branching.
* Resolution of access conflicts.
* Merging of simultaneous changes.
* Modules can be retrieved by their own revision number, system version number, or specified date.
* Efficient disk storage. PVCS uses a very intelligent difference detection technique that minimizes the amount of disk space required to store a new version.

## PVCS Maintains System Integrity

PVCS prevents system corruption that could ordinarily result from security breaks, user carelessness or malfunctions. The levels of security can be tailored to meet the needs of your project.

## The Preferred Version Control System

PVCS is now being used in personal computer software development environments at hundreds of high-technology corporations including:

| | |
|---|---|
| Hewlett-Packard | 3 Com |
| GTE Data Services | IBM |
| Sperry | Texas Instruments |
| Westinghouse Electronics | ROLM Corp. |
| Nestar Systems | The Aerospace Corp. |

**Maintains Source Code Written In ANY Language.**
Requires DOS 2.0 or higher. Compatible with the IBM PC, XT, AT and other MS-DOS PCs.

Only PVCS meets the needs of Independent Programmers and Corporations. Once you standardize on PVCS, the "Logfiles" used to track and monitor changes are interchangable between any PVCS product. You will receive full credit for your initial purchase if you upgrade to a higher-priced PVCS.

**Personal PVCS** — Offers most of the power and flexibility of the Corporate PVCS, but excludes the features necessary for multiple-programmer projects. **$149**

**Corporate PVCS** — Offers additional features to maintain source code of very large and complex projects that may involve multiple programmers. Includes "Branching" to effectively maintain code when programs evolve on multiple paths (e.g., new versions for different systems, or a new program based on an existing program). Single user. **$395**

**Network PVCS** — Extends Corporate PVCS for use on Networks. File locking and security levels can be tailored for each project. 5-Station License $1,000. Call (503) 645-1150 for pricing on Licenses for more than 5 Stations. **$1000**

**TO ORDER:** VISA/MC 1-800-547-4000. Dept. No. 306
Oregon and outside US call (503) 684-3000. Send Checks, POs to: POLYTRON Corporation, 1815 NW 169th Place, Suite 2110, Dept. 306 Beaverton, OR 97006.

POLYTRON®

High Quality Software Since 1982

Number 26 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# TURBO PASCAL GENERATOR

## THE GTP PROFESSIONAL MODEL

**Generate error-free Turbo Pascal source code for:**

**DATA BASES MENUS**
**SCREENS REPORTS**

**Build complete, working programs in minutes!**

**FEATURES**
- Indexed Data Bases
- *Multiple Screens*
- Automatic Updating
- Built-in Edits
- Retrieval Facility
- Automatic File Build
- Quick Screen Handling
- Speedy DB Access
- Context Sensitive HELP
- Full Keyboard Support

**EASY**
1. Paint-the-screen
2. Define fields & calculations
3. Generate, compile & RUN

**FLEXIBLE**
- 100% Turbo Pascal
- Modify in Pascal under Turbo's Editor
- Extensions are easy with 130 page Programmer Reference manual
- 100 documented source code routines included

NOT Copy Protected NO Royalty Fees NO Run-time Library Required

Requires: IBM PC (100% compatible) 256K RAM 2 Disk Drives
PC-DOS 2.0+ Turbo Pascal 3.0 from Borland International

**PRICE $200.00**
**CALL FOR QUANTITY DISCOUNTS**
VISA/MC — Check — Money Order — No COD or Purchase Orders
Texas Residents: add $12.25 Sales Tax
Outside US & Canada: add $25.00 Air Postage and make payment by credit card or money order in US Funds.

A E F **ALLEN, EMERSON & FRANKLIN**
P.O. Box 928
Katy, TX 77492
(713) 391-8570

Number 53 on the Reader Service Card

***If BASIC's data limit is the question . . .***

# DataSpace

***. . . is the answer***

***DataSpace*** gives MS* and GW-BASIC* users . . .

- the ability to define and access string, integer, single, and double precision data arrays
- 10 - one, two or three-dimensional arrays up to 64K each - 640K total
- array storage and retrieval from files in any disc directory
- full compatibility with interpreters or compilers

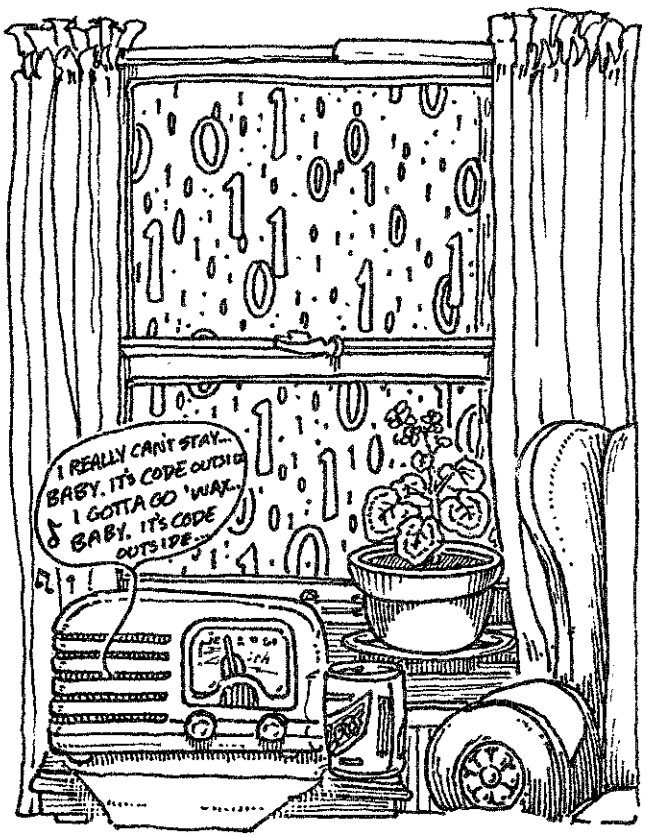
. . . all with absolutely no impact on BASIC's own data space (requires MS-DOS* 2.10 or higher)

***The only answer around for $75 . . .***

DESIGN SOLUTIONS INC.
732 Sixth Avenue
Aurora, IL 60505
**(312) 898-0651**

* MS, GW-BASIC, and MS-DOS are trademarks of Microsoft Corporation

Number 41 on the Reader Service Card




SURE IS.

# Sort RM/COBOL Files

## • • • QISORTC • • •

Finally, a sort utility for RM/COBOL.
Using efficient algorithms in assembly language making it the fastest sort on the MS-DOS market. Qisortc supports:

- RM/COBOL Version 1.5x & 2.xx files
  - Single / Dual File Indexed
  - Relative and Sequential
- ASCII, COMP-1, & COMP-3 fields
- 32 sort control fields
  - ascending / descending
- 32 record selection fields
  - EQ:NE:GT:GE:LT:LE field:constant
  - implied AND, OR may be specified
- 32 output fields or total record
- CALL from RM/COBOL, when provided subroutine is linked with runtime
- Use pathnames and return codes
- Keyboard or Parameter file
- Requires MS-DOS ver 2.xx or above
- Only $95.00 - COD, VISA, or M/C.

**Quantumn Information Systems, Inc.**
18609 76 Ave. West, Suite H
Lynnwood, WA 98037-4149
(206) 7743-7678 Telex 4941041

Number 31 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Turbo C: A First Look

by
**Michael Abrash**

*The following is a preview of Turbo C from Borland. I did not have the opportunity to actually work with the product, which is still in the Beta-test stage; this article is based on a long conversation with Brad Silverberg, Director of Engineering for Borland, and a brief demonstration of Turbo C. I should be able to try out Turbo C soon; as soon as that happens, I'll report on my findings.*

I've just returned from a long, late-night preview of Turbo C, and I'm excited. To put it briefly: With a single exception, Turbo C looks like What We've All Been Waiting For. This is not hype—it's the voice of a man who's been frustrated on a daily basis by the limitations of existing C compilers and who's suddenly seen the answer to his prayers.

All right, already, I admit it—I could be wrong. I've only seen Turbo C, I haven't used it. I saw a big file compile in a hurry, but we all know about demos. I was told some benchmark times, but there are liars, damned liars, and. . . Hey, look, for all I know there *is* no such thing as Turbo C, just a demo and some smooth patter.

Despite which, I am willing to go out on a limb and say that Turbo C *does* look like What We've All Been Waiting For: a full-featured compiler that produces excellent code in an unbelievable hurry, complete with all the extras needed to write good code for the 8088 and the IBM PC. This is a product that's got everything you'd want in a PC development environment—except a source-level debugger. There are, of course, a number of compilers that provide complete development environments; where Turbo C really stands out is in compile (and link) time, where it moves into a class all its own among full-featured C compilers.

One expects Borland to do things in nonstandard ways. After all, Turbo Pascal isn't standard by anyone's definition. Not so with Turbo C. Turbo C's developers are all C people, and C people live and die by Kernighan & Ritchie. Turbo C is full K&R (and ANSI Draft) C.

One also expects Borland to do things in unconventional ways. Turbo C is decidedly unconventional: in that, for instance, the compiler is a single program that compiles from RAM to RAM at 6000 lines or more per minute, generating code that's as fast as and more compact than that produced by any other compiler; in that it allows in-line MASM-compatible assembler code, complete with symbolic references shared with the C code; in that it allows registers to be used directly in C code; in that it offers the choice between a full DOS command-line interface and a Turbo BASIC-like environment—complete with the familiar Borland editor; in that it makes library source available (all right, that costs extra—no one's perfect); in that the floating-point arithmetic is full IEEE and yet faster (according to Borland) than the floating point arithmetic in Microsoft C.

## Turbo C *does* look like What We've All Been Waiting For

Want more? How about a set of Unix-like utilities? A linker that's much faster than MS-LINK? A price tag of less than $200—maybe less than $100?

What's that? No, I'm sorry, there are no Ginzu knives.

If Turbo C is all Borland claims, there are very few reasons why people who develop on PCs for PCs would use anything else; in fact, the lack of a source-level debugger is about the only reason I can think of right now. It's that good.

Let's take a longer look at what Turbo C has to offer.

**But Is It a Serious Development Tool?**

Given the readership of *PJ*, the first question about Turbo C has to be whether it's for the serious developer. The answer is unequivocally yes. The compiler can be run from the DOS prompt, can use all the memory in the PC, accepts standard ASCII files (so you can use your own text editor), and generates standard OBJ files that are linkable with MS-LINK. The compiler is a full K&R and current ANSI Draft implementation, with PC-specific extensions. The extensions include near, far, and huge pointers, all of the memory models of Microsoft C (including mixed models) plus one more, in-line assembler code, and direct register access in C code.

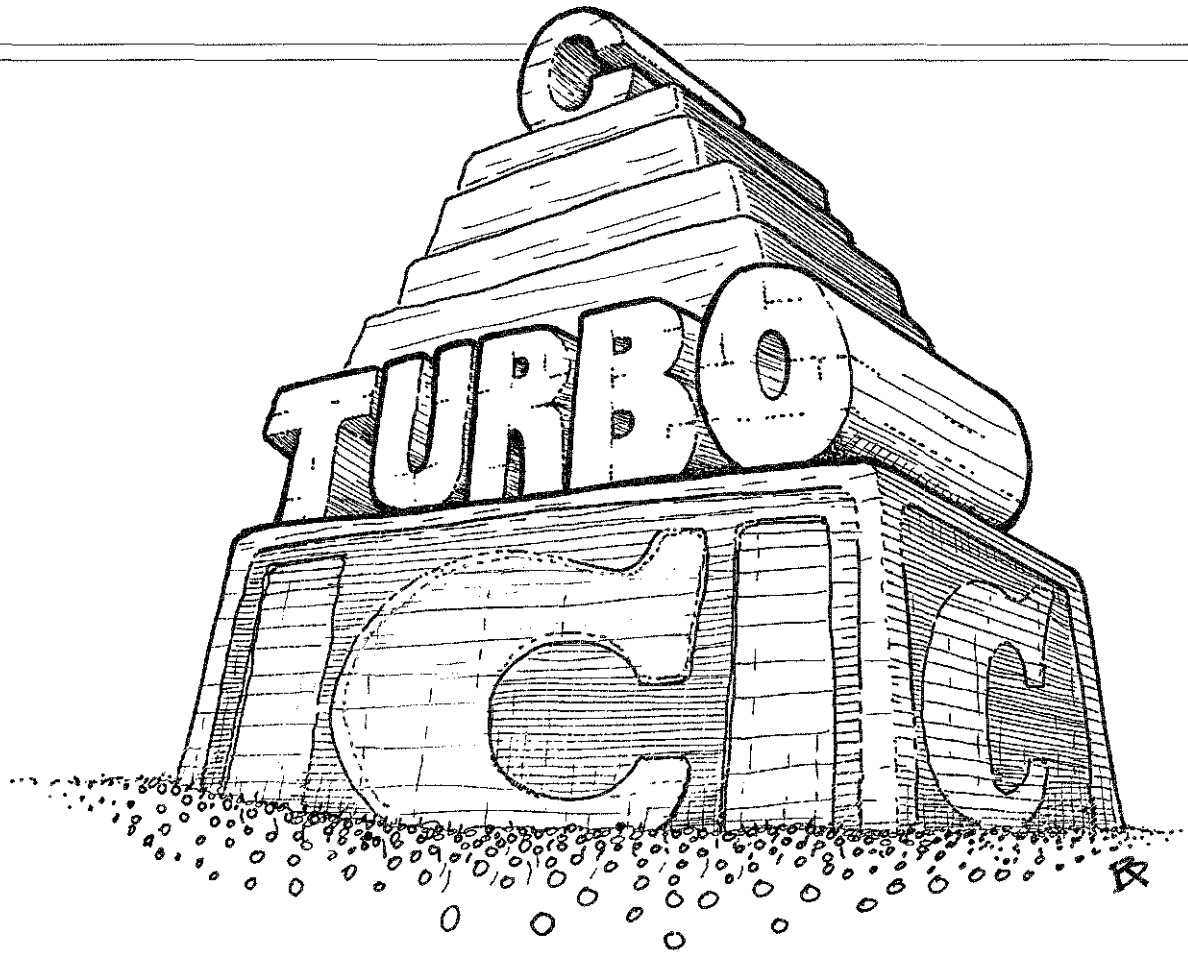
There are a number of nice touches in this compiler. For example, the programmer can specify two register variables (SI and DI) in each function. If, however, register variables are not specified, Turbo C scans the function and assigns register variables automatically. (This feature can be turned off with a command-line switch.)

Another nice touch is that all in-line library calls generated by the compiler work in all memory models. This means that near and far functions can be mixed together without the library's blowing up when a call to a library function to shift a long or perform a floating-point multiply is executed in a function that doesn't match the primary memory model. Borland also points out that developers should be able to readily address any library-related problems they may have because the library source will be available at "a Borland-style price." (Source for the startup module is included in the basic package.)

According to Borland, Turbo C's floating-point emulation library is faster than anyone else's even though it's full IEEE with 80-bit intermediate results. The floating-point routines are fully Microsoft C compatible. The standard ability to compile for



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




an 8087, for floating-point emulation, or to let the library sense whether a co-processor is installed and operate accordingly, is provided.

The in-line MASM-compatible assembler code is an even more useful feature than one might think at first. One benefit is that it is no longer necessary to incur the overhead of a function call in order to invoke assembler code from a C function. Another benefit is that even when an entire function is written in in-line assembler, C still takes care of the function exit and entry. This includes parameter handling, which can be a real nuisance in assembler, as anyone who's struggled with MASM's STRUC knows. C can also handle Pascal-style parameter passing to functions using in-line assembler, with no changes to the assembler code required. In addition to allowing easy assembler access to symbols defined in C, in-line assembly permits the C preprocessor to be used on assembler code. Among other advantages, this means that building mixed C and assembler code that can conditionally assemble/compile under more than one memory model is greatly simplified. Borland is using in-line assembler code in writing the library, which speaks well for the functionality of the in-line assembler.

Registers can be referenced directly in C statements via special names. Mixing such references into C code is not for the faint of heart, but then the use of direct register references is totally optional.

A handy feature for writing interrupt handlers is the function type "interrupt." A function of type interrupt automatically saves the state of the machine on entry and restores it on leaving, terminating with an IRET.

The compiler has an integrated lint capability and provides detailed and explicit error messages. Warnings are grouped into classes, and each warning class can be enabled or disabled individually.

Turbo C can be told to generate 80186/80286-specific code via a command line switch.

Borland says that the standard library includes everything in the Lattice and Microsoft C libraries except Bessel functions. Direct access to DOS and the BIOS is available. Far heap management routines are provided. Borland is not sure how many current third-party libraries are Turbo C compatible, but they are working with several library suppliers right now. Borland also points out (justifiably) that it's a safe bet that third-party suppliers will be in a hurry to support Turbo C. (Take a look at how many ads there are in *PJ* for Turbo Pascal add-ons.) Borland also plans to provide Turbo C toolboxes like those available for Turbo Pascal.

Turbo C can interface to Turbo Prolog. Functions to access the API in Sidekick are also included.

A variety of Unix-like tools will be included with Turbo C, including MAKE. Which other tools will be included has not yet been decided.

A Turbo C configuration file can be set up, allowing the user to customize the software. Command-line switches can then be used to augment the configuration file.

So. Turbo C is indeed for the serious developer. There's one more issue, though, and that's speed.

**Speed**

There are two aspects of speed to consider when examining a C compiler—compile speed and runtime speed. According to Borland, Turbo C's compile speed is like nothing ever seen in a full PC-based compiler. Thanks to the elimination of multiple disk-to-disk compile phases, Turbo C compiles medium and large files at a rate of at least 6000 lines per minute—5 to 7 times faster than Microsoft C. If you've ever used both a Pascal compiler and Turbo Pascal, you'll know instantly what that means. Some medium-size files actually



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

compile faster in Turbo C than do files of the same size in Turbo Pascal. I didn't have the opportunity to independently verify Borland's claims, but I saw a 2000-line file, with no comments and four include files, one of which included another file, compile in 20 seconds on a 6-MHz IBM AT. Based on my own experience with 2000-line files and Microsoft C, I'd say that a 5–to–1 compile-time ratio seems reasonable. The addition of a fast linker (more on which below) helps increase development throughput as well.

The performance of the code generated is the other aspect of compiler speed. There's more than speed at stake here, though; the criterion is code quality, which includes size as well. According to Borland, their code has proved to be at least no slower, on average, than that produced by any other compiler. They attribute this performance largely to extensive register optimization. Code size has proved to be about 10% less than that of code from other compilers.

In order to truly analyse Turbo C's code quality, we'll have to wait until the compiler is on the market. An indication in Turbo C's favor, though, is that Turbo C is written almost entirely in itself (along with some assembler), and, if the compile times are to be believed, excellent code is being generated in that case.

### The Linker

Borland claims that Turbo C would be worth buying just for the linker. That's probably true. I've been looking for a faster linker than MS-LINK, and all I've been able to find is PLINK, which costs nearly $500. According to Borland, their linker is three to ten times faster than MS-LINK. Borland's linker is a subset of MS-LINK—it's missing overlays and a few switches—but they claim that their linker is used extensively for in-house development and that no one's ever missed the omitted features. My perspective is: If the linker works for you, it is indeed worth the price of the whole Turbo C package, and if it doesn't, what the heck, it was free with the compiler.

### Requirements and Limitations

Turbo C requires a PC, AT, or compatible with 320Kb of memory (384Kb is the recommended minimum memory if the user interface, discussed below, is used) and at least one disk drive. One disk drive sounds a bit ludicrous, but because Turbo C is RAM-based, one-drive operation is possible. The RAM-based nature of Turbo C also means that floppy-based systems in general work reasonably quickly.

Because it is RAM-based, Turbo C is limited by available memory. Ah-hah, you say, here comes the bad news. Not really. Turbo C compiles each function separately, so all that has to fit in memory at any one time is the compiler (which requires about 180Kb), the symbol table, and the function being compiled (not including comments). According to Borland, the only function Turbo C ever failed to compile in 640Kb was 1600 lines long—with no comments. That's a 1600-line function, not file. As far as I could tell, there is no limit associated with file length. Systems with less memory will have correspondingly lower maximum limits on function size.

You'll have to decide for yourself whether the inability to have 1600-line functions is a problem. Personally, I've never even seen a 1000-line function, with or without comments.

No debugger is included in the Turbo C package, and Turbo C is not CODEVIEW-compatible—a serious shortcoming. I don't use CODEVIEW because it uses too much memory to let my applications run, but I know many developers who swear by it. Turbo C is SYMDEB-compatible; however, SYMDEB must be purchased from Microsoft. Borland indicated that a powerful debugging tool is under development.

Borland says that there is at least one case in which Turbo C differs from the ANSI Draft specification. They did not elaborate on what that is but claimed that the ANSI specification is wrong in that case and that both Microsoft and Lattice differ from the ANSI spec in the same way. Borland claims that in all other respects Turbo C fully meets the latest ANSI Draft specification, including the latest function prototyping form, whereby function prototypes can be specified as

```
int f(int x, char *p)
```

so that they are self-documenting. The old form,

```
int f(int, char *)
```

is also supported.

Turbo C does not generate ROMable code. Turbo C also contains no special provisions for writing memory-resident programs although a terminate-and-stay-resident library function is provided.

### The Mass Market

So far, it sounds as if Turbo C is a professional developer's tool, and it is. However, Borland didn't sell several hundred thou-

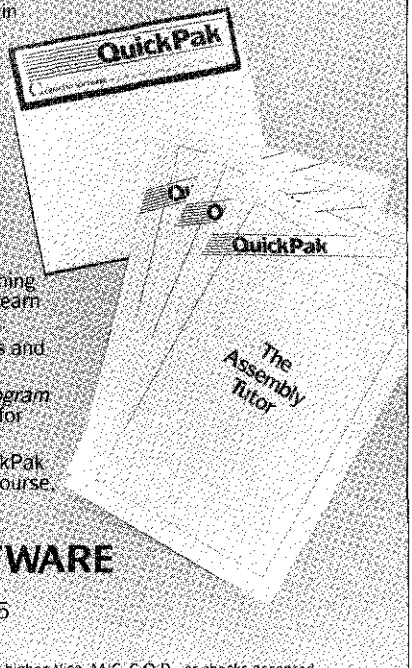

QuickBASIC just got quicker with
QuickPak

QuickPak is a superb collection of enhancements, subroutines, and instructional material designed to help you get the most out of programming in BASIC.

- Powerful assembly language routines to give your programs more speed, more power, and full access to DOS and BIOS services. SORT all or part of a string array with one command! FIND any string or sub-string within an entire array *regardless of capitalization* — accepts wildcards. READ directories into your programs from any drive or path. READ/WRITE disk sectors — create your own DOS utilities! CLEAR or SCROLL any part of the screen. MANY, many more programs included.
- Professionally written QuickBASIC routines and functions. Powerful input routines for text, dates, and numbers. Menus, scroll bars, date/time functions, and much more.
- The Assembly Tutor — a complete guide to learning assembly language *from a BASIC perspective*. Learn how to create your own routines and extensions.
- Tips and Tricks book — packed with clever ideas and techniques to help *you* be a better programmer.

You get all this, all of the source code for *every program included*, and a thirty-day money back guarantee for only $69.00.

No royalties are required for using any of the QuickPak routines in your programs. Not copy protected, of course.




by
CRESCENT SOFTWARE
64 Fort Point Street
East Norwalk, CT 06855
(203) 846-2500

QuickPak requires Microsoft QuickBASIC or BASCOM. DOS 2.0 or higher. Visa, M/C, C.O.D., or checks accepted.

Number 83 on the Reader Service Card





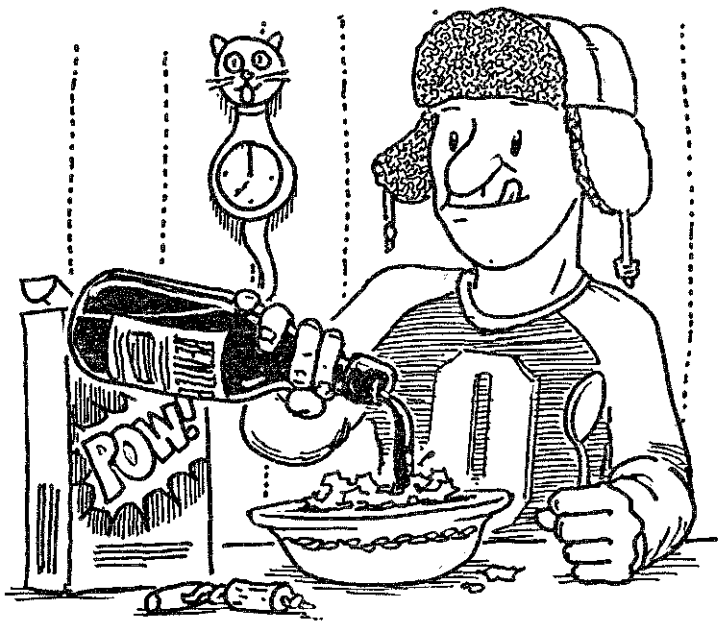
PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

sand copies of Turbo Pascal by targeting professional developers only, and the same could be said of Turbo C. Turbo C is built to appeal to non-expert programmers as well as experts. (Borland does not see Turbo C as an ideal first language—BASIC or Turbo Pascal fits that niche—but they consider it to be a logical next step.) As an alternative to operation from the DOS prompt, Turbo C comes with a user interface much like Turbo BASIC's, with command menus and multiple windows. The built-in single-file editor will be familiar to anyone who's used Turbo Pascal or Sidekick. Error messages appear in a separate window. (I did not see either the user interface or the editor in action.)

The user interface environment uses exactly the same compiler as is run from the DOS prompt. It can compile source files from disk as well as from the editor, can invoke the linker and EXEC finished programs, and even contains a stripped-down MAKE utility. In short, the user interface environment is nearly as powerful as the command-line environment, and it's self-contained.

I haven't emphasized the user interface environment because most developers I know much prefer their own editors to the one in Sidekick. Since many editors (Brief and Epsilon among them) support invoking the compiler directly from the editor, there's no real advantage for a serious developer to working in the user interface environment. For a less experienced programmer, however, the user interface could be a godsend, especially since the built-in text editor saves the cost of purchasing a third-party editor.

Borland's documentation isn't finalized yet, but it will be oriented to some extent towards less experienced programmers. The documentation will consist largely of an introduction to and tutorial on Turbo C and a discussion of Borland's extensions to the language. It will not be a comprehensive reference to the C language. Borland's perspective is that anyone can buy K&R and any of a number of reference books to learn about standard language features, so they're concentrating on getting the user up and started and on features unique to Turbo C. Also, Turbo C seems likely to




SERIAL PORT


**NEW Compiler Language:**

## PL/D is FAST

**...and comes in source**

**System Oriented:**
PL/D is designed for applications such as utilities, operating systems, and others that formerly were written partly or entirely in assembler for speed. You can access all memory and I/O. Write DOS or standalone programs.

**Fast:**
No longer do you have to trade compile speed for run time speed because PL/D gives you both. Its compile speed of 6000+ lines per minute (LPM) is the PC's fastest:

| Storage Device | CPU 8088 | Source Lines | -Compile- Time (sec) | Rate (LPM) |
|---|---|---|---|---|
| Floppy | 4.77 MHz | 2670 | 23.7 | 6759 |
| Hard Disc | 4.77 MHz | 2670 | 17.7 | 9051 |
| Ram Disc | 4.77 MHz | 2670 | 13.7 | 11693 |

PL/D is written entirely in its own language and benefits from the speed of its own generated code. Subroutine parameters are passed in registers to eliminate unnecessary stack overhead.

**Unifies Development:**
PL/D incorporates assembly level coding using the same syntax rules, listing format, and naming rules as high level parts of the program. There is no source level or run time penalty to intermix high and low level code.

**Simplifies Debug:**
PL/D eliminates the loader and its map. Source line and its locations appear on same listing line. Variables are allocated statically so nothing is lost in the stack. Subroutine location being called shows on listing line.

**Extends via Macros:**
PL/D allows you to define MACROs for extending the language. You can build tree, linked list and, ordinary linear MACROs. Compile time directives of IF#-THEN#- ELSE#-ENDIF#, WHILE#-ENDWH#, FOR#- NEXT#, and CHOOSE#-CASE#-BUT#-ENDCH# provide conditional compilation control.

**Extends via Source:**
PL/D source allows you to extend the compiler. Make edit changes and PL/D compiles a new version in significantly under 30 seconds.

**DEMO** - contains compiler with all functions except file sizes are limited. Has partial user manual on disc. **$4.95**

**OBJECT** - contains full-sized object form of compiler and complete printed 81 page user manual. **$49.95**

**COMPILER** - contains all source for PL/D compiler, 49 page internals manual and object package above. **$124.95**

NOT Copy Protected. Requires PC, XT, AT or compatible, with 2.0 MSDOS or above and 192K. Distributed on 5 1/4 inch floppy disc. Add tax in CA. 30 day guarantee.

**DAIR Computer Systems**
**3440 Kenneth Dr., Palo Alto, CA 94303**
**(415) 494-7081**





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

#1 C interpreter

C-terp

The professional C development environment

Your C compiler creates great final code . . . but as a programming tool, it's too, t o o s l o w. With C-terp you can edit, debug, and run without the wait. Nothing, but nothing, is faster for developing professional C programs.

**Choose the perfect C-terp companion for your C compiler**

| | |
|---|---|
| C-terp/Microsoft | C-terp/XENIX |
| C-terp/Lattice | C-terp/Aztec |
| C-terp/Mark Williams | C-terp/C86 |

Link in all your compiler's functions, your own functions, add-on libraries, assembly routines, and data objects. Get instant access to everything in the C-terp interactive environment.

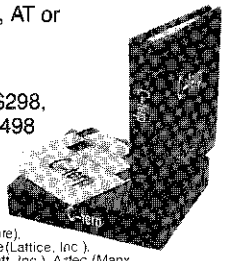
**Only C-terp offers all this and more**

- **Full K&R with common ANSI enhancements**
- **Source level interactive debugging**
- **Software paging for your big jobs**
- **Complete multi-module support**
- **Run-time pointer checking**
- **Unsurpassed reconfigurable screen editor**
- **Dual display and full graphics support**
- **Large model** ■ **Call-in**

**ORDER C-terp TODAY** (specify compiler)

C-terp runs on IBM PC, AT or compatibles.

**Price:**
MS-DOS 2.x and up - $298,
Xenix System V 286 - $498
MC, VISA, COD
30-day money-back GUARANTEE



Trademarks: C-terp (Gimpel Software), C86 (Computer Innovations), Lattice(Lattice, Inc.), Xenix, Microsoft, MS-DOS (Microsoft, Inc.), Aztec (Manx Software), Mark Williams (Mark Williams Company), IBM (International Business Machines, Inc.)

GIMPEL SOFTWARE

3207 Hogarth Lane, Collegeville, PA 19426

**(215) 584-4261**

Number 11 on the Reader Service Card

spawn books aimed at would-be programmers, just as Turbo Pascal did.

As I write this, Peter Norton's most recent column in *PC* bewails the difficulties of going from BASIC or Pascal to a professional-quality development environment. The user interface and quick response of Turbo C should ease that problem, as should the toolboxes Borland will make available.

Astute readers will note that Turbo C also provides an excellent path for programmers honing their skills. The user interface is designed for the near-beginner but uses a full-blown compiler and linker and a subset of MAKE. Substantial, multi-file programs can be developed in this environment. As the programmer becomes more skilled, the user interface can be dropped in favor of a sophisticated editor and the DOS interface, but the compiler, linker, and MAKE knowledge the programmer has acquired remain useful.

By the way, the user interface isn't necessarily useless even to a power user. One thought is that perhaps quick-and-dirty programs will finally be easier to do in C than in BASIC.

### Porting from Microsoft C

Microsoft C seems to be the leading C compiler at the moment. There are certainly other good C compilers, but Microsoft C seems to have both market share and momentum, and most developers I talk to are using it. Therefore, I'm going to briefly discuss issues of porting from Microsoft C to Turbo C.

Turbo C is nearly but not completely compatible with Microsoft C. Turbo C supports all Microsoft memory models, the Microsoft C assembler interface, near and far pointers, and the preservation of SI and DI by functions. However, static far objects in a near memory model and static near objects in a far model are not supported. That is,

```
int far farray[100];
```

is not supported, although

```
int far *farray;
```

is. Also, the group and segment names used by Borland may not be the same as those used by Microsoft (I was unable to get a definitive answer on this).

Borland claims that the port from Microsoft C to Turbo C is simple. This may well be true, and I hope to be able to report on my own experience with this port in the next issue. I would be surprised, however, if there weren't a few problems with library functions that behave slightly differently, and if there weren't some problems to resolve when linking in assembly language code.

### One Heck of a Buy—at Any Price

As you may have noticed, the topic of price hasn't come up very often in this article. That's because I think most of *PJ*'s readers are serious developers, for whom a price difference of a few hundred dollars is insignificant compared to the cost of using less-than-state-of-the-art tools.

Price is not irrelevant, however. Before I finish, I'd like to relate a short tale. It may not mean much to you, but if you're a typical *PJ* reader, I bet it'll scratch you right where it itches. Imagine that you're head of a three-person development team. All three of you use ATs and Microsoft C. You spend . . .oh, say, an hour or so each per day waiting for compiles and links. Progress is slower than you'd like. You could go to your boss and plead for three Deskpro 386s. That would cost about $18,000 and would approximately halve your compile/link time.

Alternatively, you could plead for three Turbo C packages. That would cost about $600 (maybe less) and would cut compile/link time by somewhere in the neighborhood of (conservatively) 75%.

Not a tough decision, is it? Personally, I'm hoping like heck that Turbo C is as good as claimed. Not because it's from Borland, not because it's innovative, not because it's uniquely PC-oriented, but because I need that speed. Borland has promised that I'll be able to test Turbo C soon in real production work, and when that happens, I'll let you know how it goes.

Keep your fingers crossed. □



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# HELP

123 STYLE HELP

*is at hand*

**HELP/Control**™ - an on-line help subsystem for the IBM-PC. Increases the value of your software. Save development time and money.

**HELP/Runtime.** A few simple subroutine calls add context sensitive on-line help to your application. **HELP/Runtime** includes tested interfaces for Microsoft C, Lattice C, Turbo Pascal, IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, IBM COBOL and assembler. It is distributed with demonstration programs in each language.

**HELP/Popup.** Add a powerful help system to existing applications, even in dBase or 123, without reprogramming, even without a programmer. It may be memory resident, or, installed with an application, it terminates when the application exits, releasing its memory.

**HELP/Generation.** Use your favorite editor and our concise screen definition language to build your help files. Compile them into a help system usable by either **HELP/Runtime** or **HELP/Popup**. The package includes sources for sample help files illustrating such features as full-sized or windowed screens.

**HELP/Convenience.** The screens include highlighted captions. The user selects a caption with the cursor control keys and advances to a new screen, just as with 123.

**HELP/Documentation.** A detailed manual, both on-line and printed, for the documentation writer and programmer includes instructions which may be incorporated into the user manual.

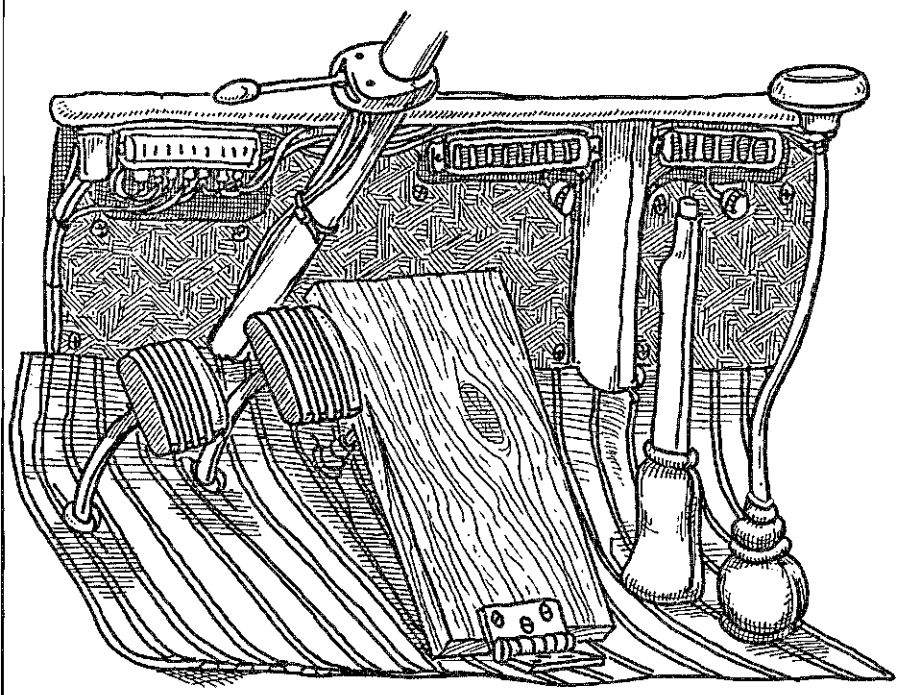
**HELP/Environment.** PC-DOS 2.0 or greater is required. **HELP/Runtime** requires approximately 9K for code and buffers for full size help screens.

**HELP/Pricing.** The complete package (software, both manuals, and demo programs) costs $125.00 and includes a royalty-free license to add **HELP/Runtime** to your applications and to make 25 copies of **HELP/Popup**. A demonstration diskette, including the on-line manual, costs $15.00. A free update to Release 1.1 is available to registered owners. To order, or for more information (including dealer, multiple-copy and site-license pricing) call MDS at 207/772-5436. We accept MasterCard and VISA.

MDS

MDS, INC., P.O. BOX 1237, PORTLAND, MAINE 04104

Number 69 on the Reader Service Card

*Well. I remember my dad had an accelerator board in a pickup of his oh, must've been 20 years ago...*


**COMBINE THE RAW POWER OF FORTH WITH THE CONVENIENCE OF CONVENTIONAL LANGUAGES**

# HS/FORTH

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & dissassemble. Not for the complete syntax checking 8086/8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility — including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/ or output stream anywhere — display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: $395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan

VISA Visa Mastercard MasterCard

**HARVARD SOFTWORKS**

PO BOX 69
SPRINGBORO, OH 45066
(513) 748-0390

Number 42 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

New Version 2.0

# Complete C Programs in Half the Time, with *Instant-C*™

You can create programs much faster with Instant-C than with conventional programming tools. How? Because Instant-C is a high-performance interpreter, there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine*, **Editor's Choice** for best C interpreter.

Powerful **source-level debugging** saves your time. Conditional breakpoints, single-stepping by statement, source code backtraces, data monitoring, and many other debugging features make it easy to wipe out bugs quickly. Direct execution of any statement or function makes testing a breeze.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy."—*The C Journal*

Instant-C checks pointer references for reasonableness, and checks that array indexes are within declared bounds. This **run-time checking** stops your program as soon as errors occur, for easiest debugging.

Not only does Instant-C help you quickly change, test, check and debug your code, but it runs your program **fast enough for real-time** applications.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—*Computer Language*

Immediate feedback and precise diagnostics make Instant-C great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes Instant-C compatible with your existing programs.

Instant-C makes all parts of the programming task as fast as possible.

"Clearly, Instant-C is the performance champion."—*PC Tech Journal*

Version 2 works with MS-DOS and PC-DOS, and has a full 31 day **money back guarantee.** Instant-C is only $495. Order today! Call or write for full information.

**Rational Systems, Inc.**
P.O. Box 480
Natick, MA 01760
(617) 653-6194

Number 20 on the Reader Service Card



**Lahey Fortran F77L Version 2.2**
*Lahey Computer Systems, Inc.*
*P.O. Box 6091*
*Incline Village, NV 89450*
*(702) 831-2500*

Just when *PJ* was completing a detailed review of F77L (Version 2.1, released in late summer 1986), Lahey sent us a new version of their fine DOS FORTRAN compiler. If you need to port scientific or technical programs from mainframe or minicomputer environments, this product may be for you. It's a fast and full FORTRAN F77 implementation that compiles from 3 to 6 times faster than the DOS FORTRAN competition. The compiler accepts a number of compiler options to make the process of porting software easier and also reads and writes large binary files, a task that plagues many other DOS compilers. The compiler has some nice diagnostic features to flag unused variables, bounds check arrays, and checks subroutine linkage. The compiler package also includes a FORTRAN source online debugger (SOLD). With SOLD, a programmer can view source code, monitor and trace the execution of a program, and modify and change variables—all without recompiling or relinking.

New features included in Version 2.2 include mainframe style listings, cross reference, and symbol tables. Version 2.2 also includes new command line options, in-line FORTRAN comments, NAMELIST input/output (a mainframe extension), easier linking, control over optimizations, and faster generated code. The debugger has also been enhanced with the command files to SOLD, save command text for later execution, search FORTRAN source files for strings, extensive display options for variables and arrays, and a single step execution of each line of source code.

**Pascal-2**
*Oregon Software, Inc.*
*6915 S.W. Macadam Ave.*
*Portland, OR*
*(503) 245-2202*
*Introductory price: $357.50*

Pascal-2 is an MS DOS Pascal development system that conforms to the level 1 ISO standard. The system is targeted mainly at programmers interested in writing applications portable to the VAX, PDP-11, 68000, and 32000 environments. It requires DOS 2 or later, a Microsoft Linker 3.0 or later, 320K minimum of memory, and a 10 Meg hard disk with at least 1.5 Meg free. The package includes utilities: a debugger, a formatter, two cross-reference generators, a macro package that simplifies calling assembly language subroutines from a Pascal program, and a BRIEF text editor interface.

The company claims a compile speed of 500 to 1500 lines a minute (depending, they say, on disk access speed). The compiler operates in five passes, and the company claims that its object code executes two or more times faster than Turbo Pascal.

Medium and large memory models are supported. Also included are some language extensions to facilitate porting Turbo Pascal programs to Pascal-2.

**386/DOS-Extender**
**386/ASM/LINK**
*Phar Lap Software*
*(617) 661-1510*

386/DOS-Extender is a software package that allows 80386 protected mode applications to run under MS DOS on a 386 based PC. Richard Smith, Phar Lap president, sees 386/DOS-Extender as a bridge product to support development of protected mode applications in the absence of a complete 386 operating system. Smith thinks that such an operating system from Microsoft won't appear until late 1987, at the earliest.

386/ASM/LINK is Phar Lap's 386 macro assembler, linker, and protected mode debugger that duplicates the 386/DOS-Extender environment.

Also available from either Phar Lap or Metaware (Santa Cruz) are 386 versions of Metaware's High-C and Professional Pascal compilers. Phar Lap President Smith feels that existing PC applications written in C or Pascal can now be recompiled under the 386 with minimal effort.

In a late-breaking note, Metaware has announced that they are about to release a 386 operating system for Hi-C.

Softguard (in Santa Cruz) is hard at work on VM 386, their virtual-machine 386 operating system. Their goal is to set up a number of virtual machines in the protected mode of the 386, each of which will be capable of running any existing PC software.

Michael Abrash has been working with these vendors and with us to bring a look at the emerging 386 environment to the pages of *PJ*.

**Turbo BASIC**
*Borland*
*(408) 438-8696*
*$99.95*

Borland claims Turbo BASIC achieves the fastest benchmarked compilation speed among commercially available BASIC compilers. Turbo BASIC includes a RAM to RAM compiler, full-screen editor, an internal linker and runtime library, and a microcalc spreadsheet complete with source code.

Turbo BASIC supports true recursion, full 8087 support, block structure, and the ability to access all of memory. Intermediate results are calculated to 80 bits of precision, and the compiler also has a 32-bit integer data type.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Inline assembly language procedures are supported, as are functions and procedures with true local, static, and global variables. Turbo BASIC is compatible with BASICA and GWBasic and supports full EGA graphics.

Turbo BASIC positions the cursor at the point of error in the source code when compile, runtime, or I/O errors occur.

Turbo BASIC is scheduled for a first quarter, 1987, release. *PJ* definitely will take a closer look, especially in comparison to QuickBASIC.

**scLASERplus, Version 1.2**
*Software Channels*
*212 King Street West*
*Toronto, Ontario*
*Canada M5H 1K5*

LASERplus is a word processor and text formatter designed to drive the HP Laserjet and Laserjet Plus. Formatting commands are imbedded in the text in ASCII. For example, formatting commands to center the word TITLE, raise the point size by two steps, select boldface type, and underline the output looks like this:

.center {large {large {bo {ul TITLE}}}}

The software also contains a deft screen capture utility that lets you import screen "pictures" into your documents in various resolutions. It also lets you draw and import icons and comes with a set of pre-defined icons.

The program can generate some sophisticated-looking output on the Laserjet. However, it is not WYSIWYG, and that means that you can end up running out 10 versions before you get it just right.

The software works pretty well, but it has rough edges. We formatted a 128-page document, for instance, and the software just stopped working at page 98 with no notice. We tried generating an index, and the index sort utility didn't seem to work—nothing that couldn't be worked around.

The formatter relies on a set of macros, and you add your own macros. That's nice although almost all the macro space in the program is already taken by the pre-defined macros. Also, the documentation is brief, and some of the macros and primitives behave in treacherous fashion. It's a package that an experienced software person can use, but it's not for someone with a low frustration level.

PC laser publishing is rapidly changing. Depending on what you have to spend for this package, it could meet some immediate needs.

**Eureka: The Solver**
*Borland*
*(408) 438-8696*
*$99.95*

Eureka solves complex problems, either directly or through iterative methods of successive approximation. The users writes an equation and sets options. The software can evaluate and display the accuracy of the solution. Problems can be solved for simple or complex variables on both sides of the equation, and up to 20 equations can be solved at a time. Eureka solves minimization and maximization problems and non-linear equations and generates tables and plots. Borland expects to find the software useful for electrical design, hydraulics, heat transfer problems, and financial calculations.

Scheduled for a first quarter, 1987, release.

**Turbo Pascal**
**Numerical Methods Toolbox**
*Borland*
*(408) 438-8696*
*$99.95*

The Toolbox is a set of applied mathematics tools, a Turbo Pascal counterpart to the IMSL and NAG Fortran libraries. The package contains ten modules with demonstration programs, including finding equation solutions, interpolations, calculus with numerical derivatives and integrals, matrix operations including inversions and determinants, Eigenvalues, differential equations, least squares approximations, and Fourier transforms.

Scheduled for a first quarter, 1987, release.

□


**DeSmet C**
*SUPPORTS*
**ROMABLE CODE**

**DeSmet C** ***$109**
Data & Stack in RAM
Code executes out of ROM
Full K&R + V7 extensions
Inline asm
Assembler, linker, librarian
Full screen editor (SEE[tm]),
8087 & S/W floating point

**DeSmet C** **$159**
**with D88 Debugger**
Set Breakpoints by line number or function name
Examine Global/Local variables by name
Show C source while debugging
Both D88 and User displays

**Large Memory** **$209**
**DeSmet C w/ D88**
32-Bit Pointers (Full Megabyte Addressing)
Fast Access of Static Data
Includes the standard compiler features mentioned above.

*Both D88 & Large Memory available as options at $50 each.
We also have a C compiler for the Mac. $150. Call for details.

*— plus —*

**Graphics** **$35**
BASICA-like (+src)

**Hacker** **$25**
Source for start-up, RAM.COM & more

**Make** **$50**
Full UNIX-level

**Tools** (+source) **$35**

**XARRAY** **$39**
Large Arrays (+source)

**C WARE**
**CORPORATION**
P.O. Box C
SUNNYVALE, CA 94087 USA
(408) 720-9696 TELEX: 358185
Street Address: 505 W. Olive, #767
VISA, MC & AMEX accepted

Number 38 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

FLASH UP YOUR PROGRAMS WITH

# POPSCREEN™

THE SCREEN GENERATOR FOR PROGRAMMERS

Create instantaneous, full color, pop-up displays and windows for your programs. Design your displays onscreen, in view, with easy, automated routines.

PopScreen automatically compiles your displays to compact .obj modules which link easily and integrally into your programs at link time. Will write displays as .asm or inline code for Assembler and Turbo Pascal.

Requires NO SATELITE DISPLAY FILES or resident loaders to accompany your C, Pascal, or Assembly programs. Your final program looks professional and is completely self-contained, in one integral file.

Dbase III and batch file programs require only one external display file for all your program's displays.

Simple to use, includes manual & tutorial. A single procedure call pops your display to screen in .015 seconds. Average display size is only 500 bytes.

PopScreen supports monochrome, cga, ega, pgc.
Assemblers: (all); Pascal: (IBM, MicroSoft, Turbo);
C: (IBM, Lattice, Microsoft); Basic with Bload; Dbase III;
Will create .com displays for batch files & applications.

**POPSCREEN, only $39.95**
BaySoft, Box 6562-J, Albany, Ca. 94706 415-527-3300

30 DAY SATISFACTION GUARANTEE

Number 87 on the Reader Service Card

dBASE Programmers

are you ready?

dB→c

"C" IS READY FOR YOU!

dB2c Offers:
- Complete implementation of dBASE III+ functions and commands in C source code.
- Interfaces for Lattice's dBC and Faircom's c-tree.
- Complete source code
- Support for the following C Compilers: Microsoft, Lattice, Aztec, DeSmet, and Instant-C.
- The freedom and peace of mind that only source code can offer.
- Works on color, monochrome, and EGA monitors.

**dB2c Toolkit $299.**

DAVID J. MARSH

Call or Write:
SOFTWARE CONNECTION, INC.
POB 712, Ely, MN 55731
(218) 365-5097

Number 93 on the Reader Service Card

## C Programmer's Toolbox

### Productivity Enhancement Tools

**CFlow** Generate a map of your program's subroutine hierarchy. Supports an unlimited number of program source listings.

**CPrint** Beautify your program listings, making them consistant and easy to read.

**PMon** Monitor your program execution and system calls, determining where the most time sensitive/critical sections are. Does not require any modification to an existing program.

***** And more.... All from one source....

**$79.95** 30 day money back guarantee
CA residents add applicabile sales tax

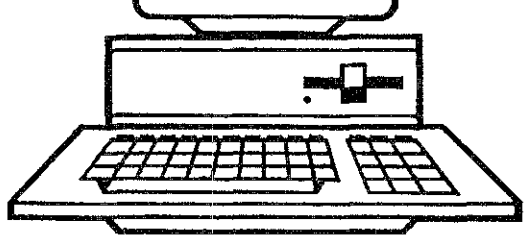
**MMC AD Systems**
Box 360845
Milpitas, California 95035
(408)-263-0781

Number 94 on the Reader Service Card

*programmer's journal*

comes to life on

# Mnematics Videotex



Mnematics Videotex brings Programmer's Journal to LIFE!!
Download source code and other information from Programmer's Journal.
Coorespond electronically with the editors and authors.
MnEmail™ Electronic Mail, Realtime™ Chat, plus much more...
We'll waive the $39.95 sign up fee, if you call and subscribe now:
1-800-322-3633 (914-365-0184 in N.Y. State)
or write:
Mnematics, Inc., Sparkill, N.Y. 10976-0019

Number 65 on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

*PJ Billboard is a directory of products and services of interest to the IBM PC programmer. The goal is to provide our readers with a wide range of information on programming products, introductions, and updates. PJ Billboard serves advertisers by providing a low cost, flexible format in which they can present their products to PJ readers.*
*The standard layout is four columns, 9 picas wide, with copy set in 7 point on 9 point leading. A PJ Billboard listing offers the advertiser a 400-character product description including company name, address, and telephone number. The listings are $89, prepaid, non-commissionable, for a one-time insertion. Listings must be received typed or legibly printed, with boldface or italic words indicated where desired. A convenient order form is available upon request. Copy deadlines are the same as the normal advertising space reservation dates, which can be found on our rate card.*

*For more information call the*
***Programmer's Journal***
*Advertising Department*
**(503) 484-2162**

**FREE YOURSELF FROM DOS BATCH**

OPAL, a rich, full-functioned batch exec language, frees you from the limited capabilities of BATCH and other primitive exec languages. OPAL, a flexible shell language offering a high degree of programming, provides a way to:

- tie applications together into one easily-used system
- create enhanced DOS batch applications
- build custom, "bullet proof" installation programs for your software applications
- produce excellent, user-friendly interfaces for your applications

Only $169.00

Write or call for more info:

**The Software Factory, Inc.**
*15301 Dallas Pkwy, Ste 750, LB 44*
*Dallas, TX 75248*
*(214) 490 - 0835*

**Curses Window Development Package** Get the Window Tool designed for serious C programmers. It's as easy to use as **printf** and even more powerful. You get all that power and Unix/Xenix compatibility too. Comprehensive manual with Quick Reference. Write or call for free information or order today (30 day money back guarantee).

Supports:
Microsoft, Lattice, and DeSmet C.

Library - $89.
Source - $289.

**Aspen Scientific**
*P.O. Box 72*
*Wheat Ridge, CO 80034-0072*
*(303) 423 - 8088*

**PCPOLL**

UNATTENDED Communications running on PC. Bisynch, SDLC, Asynch– Up to 8 lines.
Call your branch office computers, transfer files up or down, remote batch processing, Mailbox.
Scheduler provides weekend polling.
POSTPROCs process collected data.
Also
COMM. DRIVERS & SYNCHRONOUS MODEMSBOARDS.

**RTI**
*1200 Executive Dr. East, Suite 124*
*Richardson, TX 75081*
*(214) 783-4782.*

**DOS HELP**

On line help facility for DOS 3.x. Several useful utilities including backup procedures, file attribute manipulation, expanded directory listings and backup journaling. Package ships in a solid diskette carrier ideal for school or the office. Routines are available in Osborne / McGraw Hill's **DOS The Complete Reference** by Kris Jamsa.

$17.95 plus $2.50 shipping

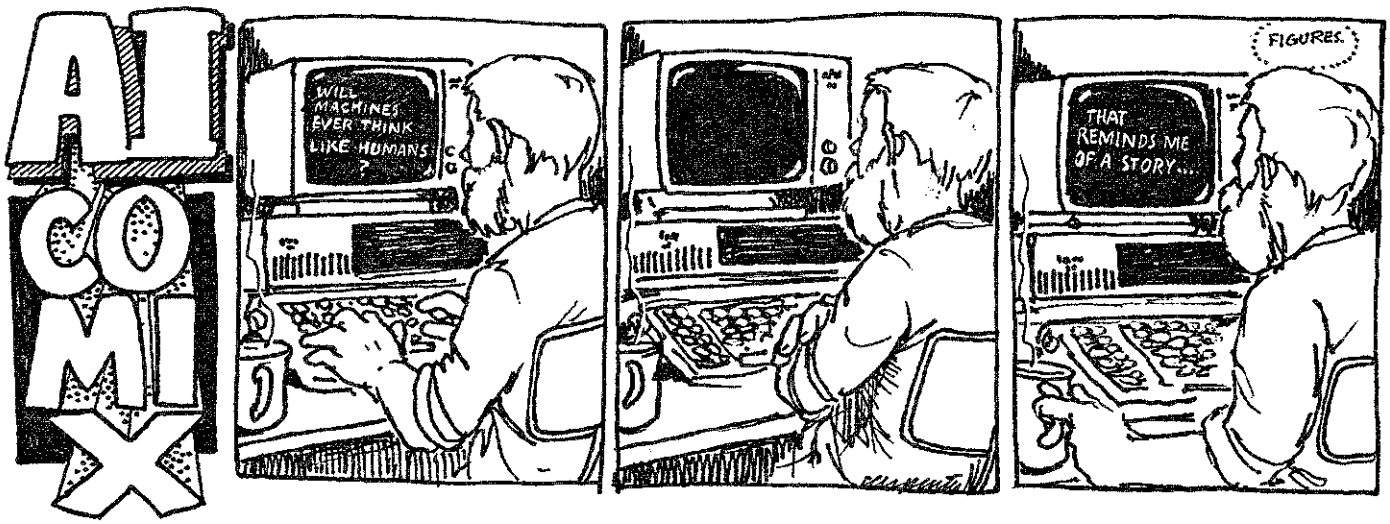
**Kris Jamsa Software, Inc.**
*Box 26031*
*Los Vegas, NV 89126*

**FREE PUBLIC DOMAIN SOFTWARE FLYER!**

IBM & Compatibles (MS-DOS) More than 4000 programs on 165 disks for only $3.50 per disk! Games, Educational, Business, Personal, Utility and Miscellaneous Software! Terrific software and service at unbelievable prices!

Send for FREE FLYER to:

**AP-JP Enterprises, Inc.**
*P.O. Box 1155*
*W. Babylon, NY 11704*






PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

**Writer Guidelines**

Submit articles for *Programmer's Journal* on disk with an accompanying hard copy to

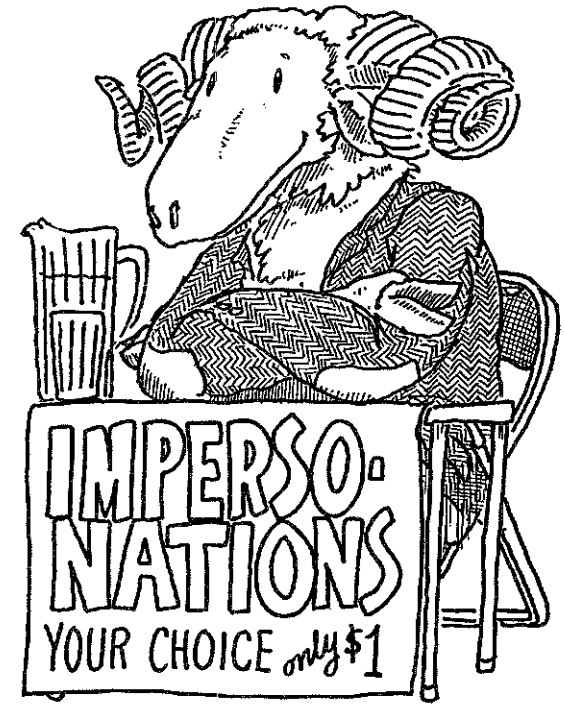
Editor
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

We like to receive ASCII text files. Mark your hardcopy for boldface, italics, etc. Don't use your word processor's printer formatting commands.

Text should be single-spaced, with unindented paragraphs, and with two carriage returns between paragraphs.

Use subheads to indicate what sections of your article are about.

Attach some information about yourself if you'd like it to appear at the end of your article. □




CUSTOM CHARACTER RAM AVAILABLE

## ADVERTISER INDEX


| Number | Advertiser Name | Page Number |
|---|---|---|
| 53 | Allen, Emerson & Franklin | 61 |
| 86 | Aspen Systems | 59 |
| 30 | Austin Code Works | 59 |
| 87 | BaySoft | 70 |
| 36 | B C Associates | 5 |
| 2 | Blaise Computing | 3 |
| 3 | Borland International | C4 |
| 38 | C-Ware | 69 |
| 64 | Chalcedony Software. | 25 |
| 56 | CompuView Products, Inc. | 9 |
| 88 | Crackerjack Software | 36 |
| * | Creative Programming Consultants | 58 |
| 83 | Crescent Software | 64 |
| 66 | Custom Software Systems | 53 |
| 5 | Dair Computer Systems | 65 |
| 41 | Design Solutions | 61 |
| 8 | Ecosoft, Inc. | 1 |
| 11 | Gimpel Software | 66 |
| 42 | Harvard Softworks | 67 |
| 57 | Hawkeye Grafix | 43 |
| 58 | Interface Group | 39 |
| 91 | King Computer Services | 19 |
| 59 | L.U.C.I.D. | 53 |
| 63 | Laboratory Microsystems, Inc. | 47 |
| 44 | Lahey Computer Systems | 52 |
| 92 | Manx Software Products | 21 |
| 69 | MDS, Inc. | 67 |
| 45 | Metaware | 57 |
| 90 | MicroHelp | 8 |
| 13 | Micromotion | 18 |
| 94 | MMC-AD | 70 |
| 65 | Mnematics Videotex | 70 |
| 95 | Mystic Canyon Software | 57 |
| 16 | Opt-Tech Data Processing | 28 |
| 102 | Oregon Software | C3 |
| 6 | Periscope Company, The | 44 |
| 96 | Phar Lap Software | 59 |
| 26 | Polytron | 60 |
| 89 | Programmer's Journal ADVERTISING INFORMATION | |
| 99 | Programmer's Journal START SUBSCRIPTION | |
| 82 | PS Business Systems | 53 |
| 101 | Quaid Software | 36 |
| 97 | Qualstar | 56 |
| 31 | Quantumn Information Systems | 61 |
| 46 | Quilt Computing | 59 |
| 93 | Software Connection | 70 |
| 20 | Rational Systems | 68 |
| 72 | Speedware | 13 |
| 85 | Sunny Hill Software | 47 |
| 98 | The Software Family | 15 |
| 35 | Turbo Source Search | 43 |
| 100 | Western Wares | 56 |
| 73 | Whitewater Group | C2 |


* This Advertiser would prefer to be contacted directly



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Digital's VAX has become the foundation for a community of development systems – PDP-11s, 68000s and PC-ATs – yet, until now, no high-performance development language spanned the range of architectures and operating systems. Now there is one. **Pascal-2**

Your VAX can become the development vehicle for PDP-11 code (RSX or RT-11 target), for MS-DOS applications, or for 68000/20 VERSAdos and UNIX systems through **Pascal-2** cross-compilers. And you can even turn the solution around and use our MS-DOS-to-VMS cross-compiler to offload your VAX. (We know what happens to VMS system response when six developers compile programs simultaneously!)

# Pascal-2™

Oregon Software's matrix of **Pascal-2 native and cross-compilers** creates a single high-level development environment to solve the programming problems faced by the world's toughest industries – aerospace, communications, robotics, process control, medicine.

**Pascal-2** is ISO standard at Level-1, the world's only certified Pascal Native and cross-compiler system. You get the portability, readability and reliability of a high-level language and structured access to the operating system or hardware when you need it. All in a rugged compiler that delivers the smallest, fastest code available.

**Dramatically improve your productivity and introduce your VAX software to the PC next door.**

Call or write OREGON SOFTWARE, INC. 6915 SW Macadam Avenue, Portland, OR 97219
(800) 367-2202 TWX: 910-464-4779 FAX: (503) 245-8449

OREGON SOFTWARE

*Real tools for real work*

**AT LAST THE PERFORMANCE IS PORTABLE**

REPRODUCED FROM BEST AVAILABLE COPY
ELECTRONIC REPRODUCTION PROHIBITED

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; IBM, PC-AT, PC-DOS, International Business Machines Corporation; MS, Microsoft Corp.;

NEW

# Turbo C®

## Turbo C: The fastest, most efficient and easy-to-use C compiler at any price

*Compilation speed is more than 7000 lines a minute, which makes anything less than Turbo C an exercise in slow motion. Expect what only Borland delivers: Quality, Speed, Power and Price.*

## Turbo C: The C compiler for amateurs and professionals

*If you're just beginning and you've "kinda wanted to learn C," now's your chance to do it the easy way. Like Turbo Pascal, Turbo C's got everything to get you going.*

*If you're already programming in C, switching to Turbo C will considerably increase your productivity and help make your programs both smaller and faster. Actually, writing in Turbo C is a highly productive and effective method—and we speak from experience. Eureka: The Solver™ and our new generation of software have been developed using Turbo C.*

## Turbo C: a complete interactive development environment

**Free MicroCalc spreadsheet with source code**

*Like Turbo Pascal® and Turbo Prolog,™ Turbo C comes with an interactive editor that will show you syntax errors right in your source code. Developing, debugging, and running a Turbo C program is a snap.*

## Turbo C: The C compiler everybody's been waiting for. Everybody but the competition

*Borland's "Quality, Speed, Power and Price" commitment isn't idle corporate chatter. The $99.95 price tag on Turbo C isn't a "typo," it's real. So if you'd like to learn C in a hurry, pick up the phone. If you're already using C, switch to Turbo C and see the difference for yourself.*

**System requirements**

*IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. One floppy drive. 320K.*

**Introductory price—good through July 1, 1987*

## Technical Specifications

- ☑ *Compiler: One-pass compiler generating linkable object modules and inline assembler. Included is Borland's high performance "Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).*
- ☑ *Interactive Editor: The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.*
- ☑ *Development Environment: A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.*
- ☑ *Links with relocatable object modules created using Borland's Turbo Prolog into a single program.*
- ☑ *ANSI C compatible.*
- ☑ *Start-up routine source code included.*
- ☑ *Both command line and integrated environment versions included.*

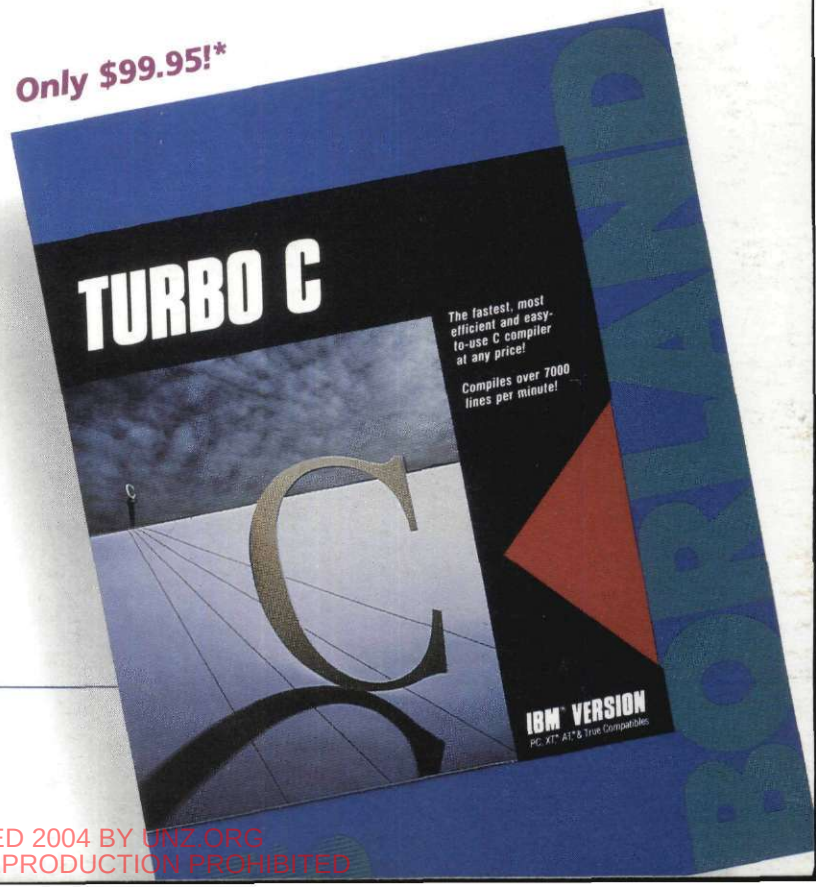
*Turbo C and Turbo Pascal are registered trademarks and Turbo Prolog and Eureka: The Solver are trademarks of Borland International, Inc. Microsoft C and MS-DOS are registered trademarks of Microsoft Corp. IBM, XT, and AT are registered trademarks of International Business Machines Corp. Copyright 1987 Borland International BI-1104*

**Sieve benchmark (25 iterations)**

| | Turbo C | Microsoft® C | Lattice C |
|---|---|---|---|
| Compile time | **3.89** | 16.37 | 13.90 |
| Compile and link time | **9.94** | 29.06 | 27.79 |
| Execution time | **5.77** | 9.51 | 13.79 |
| Object code size | **274** | 297 | 301 |
| Price | **$99.95** | $450.00 | $500.00 |

*Benchmark run on a 6 Mhz IBM AT using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51; Lattice C version 3.1 and the MS object linker version 3.05.*

**Only $99.95!***


TURBO C
The fastest, most efficient and easy-to-use C compiler at any price!
Compiles over 7000 lines per minute!
IBM VERSION
PC, XT* AT* & True Compatibles

BORLAND INTERNATIONAL

*Vive la différence*

*4585 SCOTTS VALLEY DRIVE*
*SCOTTS VALLEY, CA 95066*
*(408) 438-8400 TELEX: 172373*

TC 36

*For the dealer nearest you or to order by phone call*

***(800)255-8008***

*in CA (800) 742-1133 in Canada (800) 237-1136*

Number 3 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED